URBREATH [101139711]

Systemic Integration of Transformative Technical and Nature-based Solutions to Improve Climate Neutrality of European Cities and Regions and tackle Climate Change: the URBreath Approach

D4.7 URBREATH NBS ICT integrated solution

Project Reference No	URBREATH – 101139711
Deliverable	D4.7 URBREATH NBS ICT integrated solution - V1
Work package	WP4: URBREATH decision making framework
Type	R - Document, report
Dissemination Level	PU - Public (fully open)
Date	27/06/2025
Status	Final
Editor	Giuseppe Ciulla, Engineering Ingegneria Informatica SpA
Contributors	Rita Gaeta, Engineering Ingegneria Informatica SpA Francesco Nudo, Engineering Ingegneria Informatica SpA Giovanni Luca D'Acierno, Engineering Ingegneria Informatica SpA Ettore Etenzi, EXUS Christina Nichiforov, EXUS Chiara Savoldi, DEDA Martina Forconi, DEDA Thomas Adolphi, VCS Thanasis Dalianis, ATC Antonella Tozzi, MUN Maria Spanou, TEL Stijn Vranckx, VITO Faezeh Kazemihatami, LAT Nikos Bakalos, ICCS
Reviewers	Thomas Adolphi, VCS Maria Spanou, TEL
Document description	Integration of the URBREATH technical solution in a unique toolbox that provides a set of different digital services for end-users (e.g., digital twins, e-participation tools, dashboards and

	maps, etc.). It will also include the tool for the definition of NBS Key Performance Indicators (NBS KPI Catalogue). This deliverable is linked to T4.3 and T4.4

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
0.1	29-05-2025	Initial table of Content	Giuseppe Ciulla
0.2	30-05-2025	Content about Map Layer Manager, Integration environment, E-Participation Webb App, environment and Git Repository	Giuseppe Ciulla
0.3	04-06-2025	Content about Shadow Modelling, NBS Registry, Unified UI, 3D Map, Planner, Heat Stress Analysis, Adaptive Rainfall-Infiltration Tracking, 15 minutes city (proximity index), Dataset Catalogue, Model Catalogue, and Data sources Catalogue, Biotope Area Factor Small scale analysis, Visual Interpretable and Explainable AI	Faezeh Kazemihatami, Ettore Etenzi, Christina Nichiforov, Chiara Savoldi, Martina Forconi, Giuseppe Ciulla, Thomas Adolphi, Thanasis Dalianis
0.4	10-06-2025	Content about FROST Server, 3-30-300 Analysis, E-Participation, Mobile App, Security and User Management, KPI Manager, Publisher, Data Visualisations and Dashboards.	Chiara Savoldi, Martina Forconi, Maria Spanou, Giuseppe Ciulla, Giovanni Luca D'Acierno, Rita Gaeta, Giuseppe Ciulla, Thomas Adolphi, Antonella Tozzi
0.5	13-06-2025	Content about System Overview, Users and Roles, Structured, TimeSeries and NoSQL DB, Object Storage, GeoServer, Message Broker, Context Broker, Workflow Management and Execution, Public Transport Accessibility Analysis, 15 minutes city (proximity index), Tree growing prediction, Security and User Management, Annex A, Annex B,	Giuseppe Ciulla, Giovanni Luca D'Acierno, Francesco Nudo,Chiara Savoldi, Martina Forconi, Nikos Bakalos, Thomas Adolphi
0.6	16-06-2025	Content about Introduction, Annex C, Annex D	Giuseppe Ciulla
0.7	18-06-2025	Content consolidation	ALL
1.0	19-06-2025	Internal review	Maria Spanou, Thomas Adolphi

1.1	20-06-2025	Content consolidation	ALL
2.0	23-06-2025	Submitted to Project Coordinator for quality check	Francesco Tognoni Marcella Bonanomi
2.1	27-06-2025	Final version ready for Submission	Francesco Mureddu

Disclaimer

The URBREATH project is co-funded by the European Union under grant agreement ID 101139711. The information and views set out in this document are those of the URBREATH Consortium only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Executive Summary

The objective of the URBREATH project is to enhance climate neutrality in European cities through the adoption of Nature-Based Solutions (NBS), and tools that can support Municipalities in managing their entire life cycle (e.g. identification of problems and needs, design of potential solutions, their comparison, monitoring of the implementation phase and evaluation of generated impact).

For this purpose, the project aims to provide the "URBREATH Toolbox", a comprehensive technological framework to support municipalities in managing, implementing, and evaluating NBS interventions.

This document reports the main technical aspects of the first version of the URBREATH Toolbox; these include an update of the logical architecture of the toolbox, technical information concerning the digital services (tools) it offers in this first version (purpose, description, offered functionalities and insight about the technical implementation), main technical characteristics of the integration environment and the Git repository where the software components of the URBREATH Toolbox are made available for future reuse.

Finally, after the conclusions and the main next steps, this document reports, among the annexes, the summary of the main functionalities of the first version of the URBREATH Toolbox, the main updates of its logical architecture, the position of the URBREATH Toolbox concerning the Minimal Interoperability Mechanisms and the LDT Toolbox specifications, and the draft of the technical approach for the orchestration of analyses requested by the end users.

Table of Contents

1 INTRODUCTION 10

1.1 PURPOSE AND SCOPE 10
1.2 APPROACH FOR WORK PACKAGE AND RELATION TO OTHER WORK PACKAGES AND DELIVERABLES 10
1.3 METHODOLOGY AND STRUCTURE OF THE DELIVERABLE 10

2 SYSTEM OVERVIEW 12

2.1 HIGH-LEVEL DESCRIPTION OF THE TOOLBOX 12

3 USERS AND ROLES 14

4 DIGITAL SERVICES PROVIDED 16

4.1 DATA STORAGE 16
4.1.1 FROST STA Server 16
4.1.2 Structured, Timeseries and NoSQL DB 17
4.1.3 Object Storage 19
4.1.4 GeoServer 21
4.2 DATA CONNECTORS 22
4.2.1 Data Catalogues Connectors 22
4.2.2 IoT Connectors 24
4.2.3 IT Platform Connectors 25
4.2.4 Data Repositories Connectors 27
4.2.5 Data Spaces Connector 28
4.3 COMPONENTS COMMUNICATION 29
4.3.1 Message Broker 29
4.3.2 Context Broker 31
4.4 PROCESS ORCHESTRATION 34
4.4.1 Workflows Management and Execution 34
4.5 DATA ANALYSIS AND PROCESSING 36
4.5.1 Heat Stress Analysis 36
4.5.2 Shadow Modelling 37
4.5.3 Adaptive Rainfall-Infiltration Tracking 40
4.5.4 Biotope Area Factor (small scale analysis) 45
4.5.5 3-30-300 Analysis 46
4.5.6 Public Transport Accessibility Analysis 49
4.5.7 15 minutes city (proximity index) 50
4.5.8 Visual Interpretable and Explainable AI 53
4.5.9 Tree growing prediction 58
4.6 CITY MONITORING 61
4.6.1 KPI Manager 61
4.7 NBS PLANNING 69
4.7.1 NBS Registry 69
4.8 DATA DISCOVERY AND SHARING 74
4.8.1 Datasets Catalogue, Models/Tools Catalogue, and Data sources Catalogue 74
4.8.2 GeoNetwork 83
4.8.3 Publisher 85
4.9 CITY DATA VISUALISATION 88
4.9.1 Unified UI 88
4.9.2 Data Visualisations and Dashboards 91
4.9.3 Map Layers Manager 95

4.9.4 3D Map *97*
4.9.5 Planner *99*
4.10 E-PARTICIPATION 100
4.10.1 Web App *101*
4.10.2 Mobile App *117*
4.11 SECURITY AND USER MANAGEMENT 122
5 INTEGRATION ENVIRONMENT 124
6 GIT REPOSITORY 126
7 CONCLUSIONS AND NEXT STEPS 127
8 REFERENCES 129
9 ANNEX A - URBREATH TOOLBOX 1ST RELEASE FUNCTIONALITIES 130
10 ANNEX B - URBREATH TOOLBOX LOGICAL ARCHITECTURE MAIN UPDATES 133
11 ANNEX C - URBREATH TOOLBOX POSITION TOWARDS MINIMAL INTEROPERABILITY MECHANISMS AND LDT TOOLBOX SPECIFICATIONS 134
12 ANNEX D – OVERVIEW OF THE ANALYSIS ORCHESTRATION PROCESS 147

List of Figures

Figure 1: Logical architecture of the URBREATH Toolbox 13
Figure 2: Object Storage - MinIO 20
Figure 3: GeoServer 22
Figure 4: IT Platform Connector 25
Figure 5: TRUE Connector components 29
Figure 6: Message Broker (Apache Kafka) 31
Figure 7: Context Broker (Orion-LD) 33
Figure 8: Workflows Management and Execution (Apache Airflow) 34
Figure 9: Shadow modelling - implementation components 40
Figure 10: Adaptive Rainfall-Infiltration Tracking - Architecture 44
Figure 11: Biotope Area Factor (small scale analysis) - Sequence diagram 46
Figure 12: Prototyped version of the VIE-AI tool interface, highlighting the *Individual Predictions* tab. 54
Figure 13: VIE-AI System Architecture Diagram 57
Figure 14: Tree growing prediction - sequence diagram 60
Figure 15: Tree growing prediction – example of report 1 of 3 60
Figure 16: Tree growing prediction – example of report 2 of 3 61
Figure 17: Tree growing prediction – example of report 3 of 3 61
Figure 18: KPI Manager – Internal components 63
Figure 19: KPI Manager - Creation of a measure 65
Figure 20: KPI Manager - Acquisition of measurements 65
Figure 21: KPI Manager - Title and description of a KPI 66
Figure 22: KPI Manager - Selecting the measure to use 67
Figure 23:KPI Manager - Insert formula, schedule and target 67
Figure 24:KPI Manager - Select Chart Type 68
Figure 25: KPI Manager – Save the KPI 68

Figure 26: NBS Registry – Search page 71
Figure 27: NBS Registry – Details page 72
Figure 28: NBS Registry – Create new NBS 73
Figure 29: NBS Registry architecture 74
Figure 30: Datasets, Models/Tools, and Data sources Catalogues – Search UI 76
Figure 31: Datasets, Models/Tools, and Data sources Catalogues – Details of results 77
Figure 32: Datasets, Models/Tools, and Data sources Catalogues – Management 78
Figure 33: Datasets, Models/Tools, and Data sources Catalogues – Add a new resource 79
Figure 34: Datasets, Models/Tools, and Data sources Catalogues – Add general information 80
Figure 35: Datasets Catalogue – Federate an external catalogue 81
Figure 36: Datasets Catalogue – Management of federated catalogues 82
Figure 37: Datasets Catalogue – Action on a federated catalogue 82
Figure 38: Datasets, Models/Tools, and Data sources Catalogues – Components for managing new resources 83
Figure 39: GeoNetwork 84
Figure 40: Unified UI 89
Figure 41: NBS Stories 90
Figure 42: Unified UI - Architecture 91
Figure 43: Data Visualisations and Dashboards 94
Figure 44: Map Layers Manager 95
Figure 45: Map Layers Manager – Architecture 97
Figure 46: 3D Map 98
Figure 47: E-Participation tools - Main components diagram 101
Figure 48: E-Participation tools Web App - Integration of the Dataset Catalogue - Search 103
Figure 49: E-Participation tools Web App - Integration of the Dataset Catalogue – List of results 103
Figure 50: E-Participation tools Web App - Integration of the Dataset Catalogue – Details of results 104
Figure 51: E-Participation tools Web App - Integration of the Dataset Catalogue – Starred result 105
Figure 52: E-Participation tools Web App - Integration of the Dataset Catalogue – Add reference to starred results (1 of 4) 105
Figure 53: E-Participation tools Web App - Integration of the Dataset Catalogue – Add reference to starred results (2 of 4) 105
Figure 54: E-Participation tools Web App - Integration of the Dataset Catalogue – Add reference to starred results (3 of 4) 106
Figure 55: E-Participation tools Web App - Integration of the Dataset Catalogue – Add reference to starred results (4 of 4) 106
Figure 56: E-Participation tools Web App - Integration of the 3D Map – Add 3D Map to a participatory space 107
Figure 57: E-Participation tools Web App - Integration of the 3D Map – Management of 3D Maps 107
Figure 58: E-Participation tools Web App - Integration of the 3D Map – Register a 3D Map 108
Figure 59: E-Participation tools Web App - Integration of the 3D Map – Add attachments to a 3D Map 109
Figure 60: E-Participation tools Web App - Integration of the 3D Map – Explore 3D Maps of a participatory space 109
Figure 61: E-Participation tools Web App - Integration of the 3D Map – Visualise a 3D Map - 1 of 2 110
Figure 62: E-Participation tools Web App - Integration of the 3D Map – Visualise a 3D Map - 2 of 2 111
Figure 63: E-Participation tools Web App - Integration of the 3D Map – Visualise attachments of a 3D Map - 1 of 2 111
Figure 64: E-Participation tools Web App - Integration of the 3D Map – Visualise attachments of a 3D Map - 2 of 2 111
Figure 65: E-Participation tools Web App - Integration of the 3D Map – Comments management 112

Figure 66: E-Participation tools Web App - Integration of the whiteboards – Add whiteboards to a participatory space 113
Figure 67: E-Participation tools Web App - Integration of the whiteboards – Management of whiteboards 113
Figure 68: E-Participation tools Web App - Integration of the whiteboards – Register a whiteboards 114
Figure 69: E-Participation tools Web App - Integration of the Security and User Management – Login page - 1 of 2 115
Figure 70: E-Participation tools Web App - Integration of the Security and User Management – Login page - 2 of 2 115
Figure 71: E-Participation tools Web App - Integration of the Dataset Catalogue – Schema 116
Figure 72: E-Participation tools Mobile App – Home page 119
Figure 73: E-Participation tools Mobile App - Ideas/Proposals on the Map 120
Figure 74: E-Participation tools Mobile App – Near me 121
Figure 75: URBREATH Git Repository 126
Figure 76: MIMs Plus (version 7.5) Framework Overview 134
Figure 77: LDT Toolbox specifications Reference Architecture 142
Figure 78: Draft design of the process for analysis orchestration 148

List of Tables

Table 1: Personas and roles 14
Table 2: URBREATH Toolbox components databases 18
Table 3: Data Catalogues Connectors 23
Table 4: IoT Agents 24
Table 5: IT Connector functional blocks 25
Table 6: Data Repository Connectors 27
Table 7: Workflows Management and Execution – Main features 35
Table 8: Shadow Modelling Results 37
Table 9: 15 minutes city – technical background 53
Table 10: Mobile App – Technology stack 122
Table 11: URBREATH Toolbox position with respect to MIMs Plus version 7.0 135
Table 12: URBREATH Toolbox position with respect to MIMs Plus version 7.5 138
Table 13: URBREATH Toolbox position with respect to LDT Toolbox specifications 142
Table 14: Analysis draft data model 148
Table 15: Analysis management draft REST APIs 148
Table 16: Request draft data model 149
Table 17: Request management draft REST APIs 149

List of Terms and Abbreviations

Abbreviation	Definition
AOI	Area of Interest
API	Application Programming Interface
BAF	Biotope Area Factor
CKAN	Comprehensive Knowledge Archive Network
CRUD	Create Read Update Delete
CSW	Catalog Service for the Web
DACT-AP	Data Catalog Vocabulary – Application Profile
DAG	Directed Acyclic Graph
DSM	Digital Surface Model
FROST Server	Fraunhofer Open Source SensorThings Server
IDS	International Data Spaces
JVM	Java Virtual Machine
KML	Keyhole Markup Language
KPI	Key Performance Indicator
LDT	Local Digital Twin
Local Interpretable Model-agnostic Explanations	LIME
LoRaWAN	Long Range Wide Area Network
LST	Land Surface Temperature
M2M	Machine to Machine
MFA	Multi-Factor Authentication
MQTT	Message Queuing Telemetry Transport
NBS	Nature Based Solution
NGSI	Next Generation Service Interfaces
OGC	Open Geospatial Consortium
OTP	One Time Password
PDP	Partial Dependence Plot
PoI	Point of Interest
RBAC	Role-based access control
RDBMS	Relational Database Management System
REST	Representational state transfer
SHAP	SHapley Additive exPlanations
SSO	Single Sign-On
TLS	Transport Layer Security
UHIs	Urban Heat Islands
VIE-AI	Visual Interpretable and Explainable AI tool
XAI	EXplainable AI
WKT	Well-known text

1 Introduction

1.1 Purpose and Scope

This document reports and describes the first version of the integrated URBREATH Toolbox, which represents the interoperable collection of IT solutions made available for supporting the pilots' activities.

The main objective of the URBREATH Toolbox is to offer a suite of software tools enabling and building a comprehensive technological framework to support the Municipalities in the implementation of the different steps through which NBS interventions are managed (e.g. engagement of stakeholders, identification of local needs, design of possible NBSs, simulation of their potential effects, selection of the solution to be implemented, monitoring of the implementation, evaluation of the generated impacts, etc.).

This document reports the technical tools included in the first version of the URBREATH Toolbox, their functionalities and technical implementation.

1.2 Approach for Work Package and Relation to other Work Packages and Deliverables

This deliverable is the result of close collaboration within the URBREATH project, which extends beyond WP4 (to which it belongs).

Three work packages contributed their activities to the creation of the project knowledge that underpins this deliverable.

Besides being responsible for this deliverable, WP4 contributed by setting the path of the overall technical approach of the URBREATH Toolbox, as well as the implementation of software components such as the 3D Map, KPI Manager, and e-Participation tools.

In parallel, WP3 provided the components implementing analysis and simulations, as well as data management.

Finally, WP5, which aims to put into action technological solutions of the URBREATH Toolbox, offered relevant input by establishing a feedback loop with direct contact with the cities.

1.3 Methodology and Structure of the Deliverable

Based on the results documented in D2.4 and D2.5 and input from WP5, WP3 and WP4, a series of functionalities and tools were selected for inclusion in the first version of the URBREATH Toolbox. This selection considered the end user's perspective and the macro area of the URBREATH architecture.
In addition to software implementation, technical activities focused on the definition of communication and orchestration processes between key components of the URBREATH Toolbox.

The overall results are reported in this document and organised in the following structure.

- Section 2 reports an overview of the updated logical architecture of the URBREATH Toolbox
- Section 3 briefly reports the main users of the URBREATH Toolbox (following personas identified in D2.5) and the role they can play within the toolbox.
- Section 4 reports the digital services that are part of the first version of the URBREATH Toolbox.
- Section 5 briefly reports the technical characteristics of the digital infrastructure employed for the implementation and integration of the software components that are part of the URBREATH Toolbox.
- Section 6 briefly reports on the URBREATH Git Repository, where the source code of open-source components of the toolbox and technical information are made available to the public.
- Section 7 reports conclusions and next steps.
- Section 8 reports references.
- Section 9 summarises the features for the end users of the first version of the URBREATH Toolbox.
- Section 10 summarises the main updates of the logical architecture of the URBREATH Toolbox.
- Section 11 summarises the positions of the URBREATH Toolbox towards the Minimal Interoperability Mechanisms and the LDT Toolbox specification
- Section 12 briefly reports the draft process for the orchestration of analyses requested by the end users.

2 System Overview

2.1 High-Level Description of the Toolbox

The URBREATH Toolbox is designed to provide support functionalities for collaborative projects focused on Nature Based Solutions, that require the interactions between interdisciplinary experts and heterogeneous stakeholders. To enable effective collaboration among them, the URBREATH Toolbox provides a unified framework that facilitates coordination, data integration, and sustainable project outcomes. The URBREATH Toolbox leverages a logical architecture designed around the following key characteristics.

- Promotion of cross-disciplinary collaboration
- Centralise discovery and access to heterogeneous and scattered data.
- Adaptiveness and flexibility, to allow the reuse of the toolbox over time for different projects.
- Facilitate stakeholder engagement and communication, ensuring that local voices are heard and that projects meet local needs
- Ability to perform simulation and analysis, enabling stakeholders to model and compare different scenarios of interventions and assess the potential impacts of NBS projects.
- Define, track, and report on key performance indicators.
- Ability to perform data visualisation, transforming complex data into understandable visual representations such as charts, graphs, and 2D/3D maps.

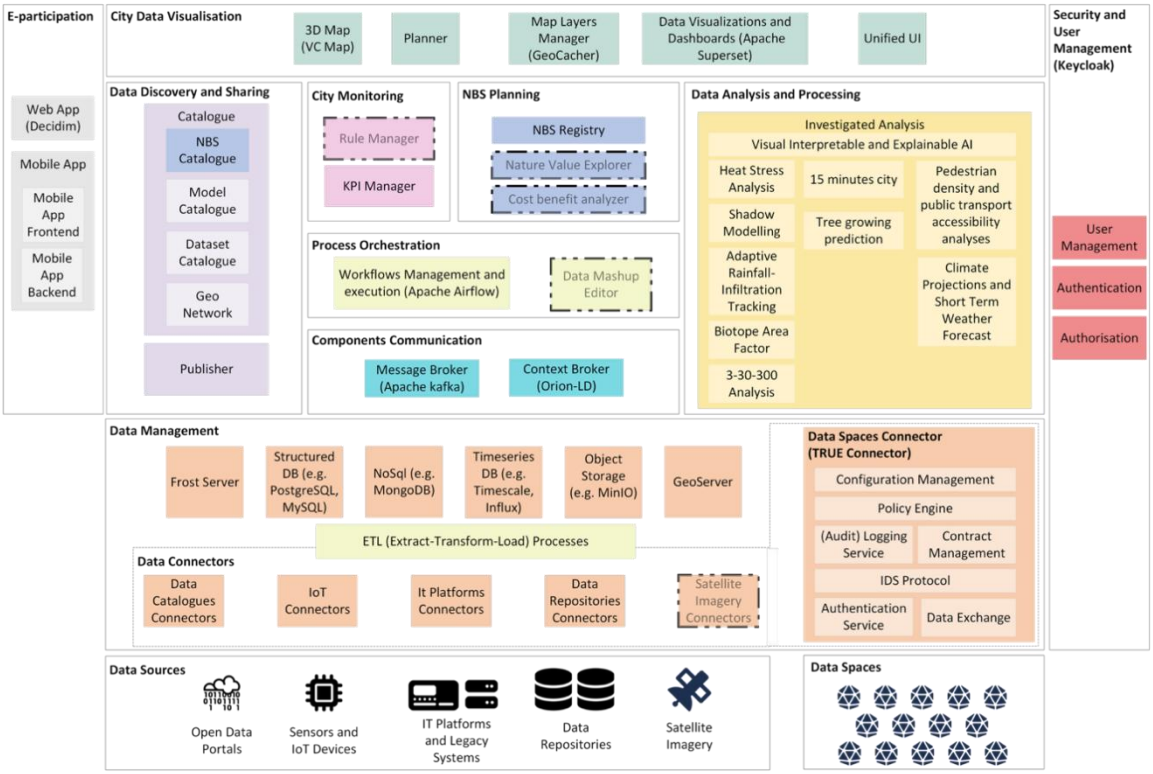
These characteristics are developed within a logical architecture organised into ten macro areas (D2.5), summarised in the following and depicted in Figure 1[1].

- **Data Discovery and Sharing** offers a unique and uniform access point to discover the whole set of information managed through the URBREATH Toolbox (i.e. datasets collected from connected systems and produced by the toolbox, information about available models, tools, data sources, etc.).
- **City Data Visualisation** offers the UIs of the components of the URBREATH Toolbox which require users' interaction (e.g. for data visualisation).
- **City Monitoring** includes functionalities to monitor parameters of interest in the city (i.e. KPIs).
- **Data Analysis and Processing** includes specific data analytics and simulations, identified according to the needs and purposes of the URBREATH pilots.
- **NBS Planning** offers functionalities to support decision makers, planners, etc. in investigating, designing, and planning NBSs.

[1] Section ABC summarises the main changes with respect to the initial logical architecture of the URBREATH Toolbox reported in the Deliverable "D2.5 URBREATH platform requirements".

- **E-Participation** supports the engagement of stakeholders and their interaction within participatory process for planning, monitoring and evaluating interventions such as NBSs projects.
- **Data Management** provides functionalities to collect and manage data from heterogeneous data sources, and to interoperate with data spaces.
- **Components Communication** offers functionalities enabling intercomponent communication channels that allow the other components of the URBREATH Toolbox to interact.
- **Process Orchestration** allows to define execute and orchestrate complex processes that involves different components of the URBREATH Toolbox.
- **Security**: this area includes the functionalities that allow to manage users, their role, access permissions, as well as the user authentication and authorisation.

Figure 1: Logical architecture of the URBREATH Toolbox



Each macro area offers a series of digital services (detailed in section 4) that cover a specific set of homogeneous functionalities. The ten macro areas are complementary with respect to each other and together offer the complete set of capabilities of the URBREATH Toolbox. It is important to underline that, despite the mutual complementarity of the macro areas, the URBREATH Toolbox is conceived as modular, offering the opportunity to select single components of interest for local deployment or "slices" of the toolbox. Thus, future reusability and replicability opportunities across, beyond the URBREATH project, are ensured.

3 Users and roles

The URBREATH Toolbox is designed to serve a diverse group of users ranging from city officials and domain experts to community members and visitors. To ensure ease of use for them, the application offers different experiences depending on the role and level of access of each user. The typical journey through the platform can be broken down to three kinds of users: Admin, Logged User, Data Scientist and the Guest User.

Admins are representative of the Municipality with expertise, entitled to manage the platform. A logged user is any stakeholder interested in contributing with comments and new content, required to login to operate. Data Scientists are invited users with expertise in different fields. Guest users are instead only interested in accessing public content within the platform with no need for registration.

For the sake of clarity, a direct correlation between personas defined in D2.5 and the roles they can play is summarised in Table 1.

Table 1: Personas and roles

Persona	Municipality Role	Objective	Application Role
Arantxa Sánchez	Urban planner and policy maker	Design Innovative NBSs to satisfy specific needs	Data Scientist
Luigi Riva	Local Government Official	Deliver effective and legally compliant solutions	Admin
Kim Clijsters	Environmental Scientist, Technology Developer, and Data Analyst	Ensures that NBSs are environmentally beneficial and efficient	Data Scientist
Gheorghe Hagi	Real Estate Developer and Investor	Achieving a balance between financial return and social impact through NBSs	Logged in
Emilio Butragueño	Community Leader	Foster a participatory planning process and maintain strong community support	Admin
Elizabeth Bennet	Dedicated Citizen	Contribute to community growth	Logged in

The URBREATH toolbox ensures that each audience whether managing, contributing, or simply exploring, has a clear and intentional experience aligned with their goals.

The functionalities provided to the different roles are managed by Security and User Management area, defined in section 4.12 of this document (that is based on Keycloak, which guarantees needed features such as role-based access control and session management).

Admins are typically municipal representatives and community leaders, charged to manage content published on the platform. After logging in with secure credentials, they end up on the URBREATH toolbox dashboard where they can oversee all activities.

Registered users, such as stakeholders or municipal partners, register to participate more actively in decision-making processes. In fact, by logging in, they can explore content, post comments, and submit their own experiences, projects, or ideas.

Arantxa Sánchez's persona, as an urban planner and policy maker, must be able to generate simulations involving her plans and policies. The same must be true for Kim Clijsters and all other partners involved in technical tasks. To achieve this, the **role** Data Scientist guarantees their specific privileges.

The role "**Guest users**" don't need to register to access public content. They can browse stories, maps, datasets, and reports freely. However, they can't comment or contribute. Throughout their visit, they're encouraged to sign up for full access and community engagement. Even without an account, guests can gain valuable insights and learn from the shared experiences available on the platform.

4 Digital Services Provided

This section summarises the digital services (tools) that are part of the first version of the URBREATH Toolbox, describing their purpose, offered functionalities and main technical details.

4.1 Data Storage

4.1.1 FROST STA Server

4.1.1.1 Description and Purpose

The FROST®-Server[2] (Fraunhofer Open Source SensorThings Server) is an open-source implementation of the OGC SensorThings API[3] developed by Fraunhofer IOSB[4]. It provides a high-performance, resource-efficient platform for storing and accessing IoT sensor data via a standardized RESTful interface. Designed for use in both research and commercial applications, FROST® supports real-time scenarios such as smart cities, environmental monitoring, and emergency management. It is also recognized by the European Commission as a best practice for sharing measurement data under INSPIRE guidelines. The data model implemented by FROST® is compliant with OGC standards[5].

4.1.1.2 Key Features and Functionality

The FROST®-Server offers a range of key features and functionalities:

- The FROST®-Server is freely available as open-source software.
- The FROST®-Server implements the OGC SensorThings API[6], ensuring interoperability and adherence to international standards.
- It is designed to deliver fast data processing with minimal resource consumption.
- It provides a standardized RESTful API for seamless integration with diverse applications and systems.
- It supports real-time access and management of IoT sensor data, suitable for dynamic scenarios such as smart cities, environmental monitoring, and emergency response.
- It is recommended by the European Commission as a "Good Practice" for sharing measurement data in compliance with INSPIRE guidelines.
- It ensures that sensor data remains fresh, accessible, and reliable for continuous use.

4.1.1.3 Technical implementation

The FROST Server comes in three packages, and thus also in three docker images:

[2] https://github.com/FraunhoferIOSB/FROST-Server
[3] https://github.com/opengeospatial/sensorthings
[4] Institute of Optronics, System Technologies and Image Exploitation; https://www.iosb.fraunhofer.de/en.html
[5] https://fraunhoferiosb.github.io/FROST-Server/sensorthingsapi/requestingData/STA-Data-Model.html
[6] https://www.ogc.org/standards/sensorthings/

1. The all-in-one package[7] contains both the HTTP and the MQTT parts of the FROST-Server. The HTTP and MQTT parts can directly communicate, and they run in the same JVM (Java Virtual Machine[8]), with minimal delay between updates and MQTT message delivery.
2. The HTTP only package[9] contains the HTTP part of the FROST-server. It should be used together with the MQTT only package, to have an arbitrary number of HTTP and MQTT components, connecting to the same database, and communicating among each other through the message bus.
3. The MQTT only package[10] it contains the MQTT part of the FROST-server. To support multiple HTTP and MQTT instances, a message bus was introduced.

The FROST Server can be run in Tomcat or as a docker image[11]; A PostgreSQL server with PostGIS extensions is also required[12].

PostgreSQL is a powerful, open-source object-relational database management system known for its reliability, rich feature set, and high performance. The official documentation offers comprehensive guidance on installation and usage, and the open-source community provides numerous resources for learning and exploring its capabilities.

PostGIS is an extension for PostgreSQL that adds support for geographic data. It enables the storage, indexing, and querying of spatial data within the database, making PostgreSQL a powerful tool for geographic information systems (GIS).

4.1.2 Structured, Timeseries and NoSQL DB

Given the diverse and voluminous nature of the data it processes, ranging from sensor time series to spatial and semantic metadata, selecting appropriate data storage systems was a critical design consideration. This chapter details the rationale, architecture, and implementation of the data storage systems adopted in the URBREATH toolbox. Three principal categories: structured relational databases, time-series databases, and NoSQL databases, analysing their roles, benefits and limits.

4.1.2.1 Structured Database

Structured databases, particularly relational database management systems (RDBMS), are used in URBREATH to handle tabular data with well-defined schemas.

Main strengths of these systems are guaranteed reliability and consistency of critical metadata, and schema enforcement (enhancement of data integrity and enabling of complex queries). On the other

[7] https://hub.docker.com/r/fraunhoferiosb/frost-server/
[8] https://www.java.com/en/download/help/whatis_java.html
[9] https://hub.docker.com/r/fraunhoferiosb/frost-server-http/
[10] https://hub.docker.com/r/fraunhoferiosb/frost-server-mqtt/
[11] https://fraunhoferiosb.github.io/FROST-Server/deployment/docker.html
[12] https://fraunhoferiosb.github.io/FROST Server/deployment/architecture packages.html

hand, these kinds of DBs are less suitable for high-velocity data ingestion and large-scale temporal data, which led to the integration of more specialized systems for time-series data.

Two structured databases are used in the system: as the unified data backend for the dashboards the chosen one is **PostgreSQL**, a relational database with spatial extensions (**PostGIS**), used for storing structured metadata and supporting complex queries and geospatial operations. And **MySQL**, used in legacy modules or for lightweight structured data storage where spatial or advanced querying features are not required, offering broad compatibility and ease of deployment.

4.1.2.2 Timeseries Database

Sensors and meteorological instruments generate high-frequency, timestamped measurements. Managing this volume of data efficiently required a time-series database optimized for fast writes and time-based queries.

In this context the used one is **InfluxDB**, optimized for high-frequency sensor data that enables fast data ingestion, time-based queries, and efficient long-term storage through retention policies.

4.1.2.3 NoSQL Database

The URBREATH toolbox deals with semi-structured and unstructured data. To handle this flexibility, a NoSQL document-oriented database was necessary. Main strengths of these structures are flexible schemas that supports evolving data structures without costly migrations, powerful querying on nested documents ideal for storing complex configuration data and model results, and horizontal scalability to facilitate performance tuning and high availability for distributed deployments.

For these purposes **MongoDB** was chosen, a NoSQL document database used to store semi-structured and unstructured data such as model outputs, configuration files, and user-generated content. Its flexible schema supports evolving data formats.

Table 2 summarises the main databases used by the URBREATH Toolbox components.

Table 2: URBREATH Toolbox components databases

URBREATH component	Purpose	Database
GeoCacher	Generated layers data	MongoDB
Context Broker (Orion-LD)	Context data (NGSI-LD entities)	MongoDB
E-Participation Web App (Decidim)	Data for digital participation (proposals, votes, etc.)	PostgreSQL
E-Participation Mobile App	Data for application mobile	MongoDB
KPI Manager	Definition of KPIs (description, associated data sources, formula, etc.)	Postgresql
	Calculated values of KPIs	InfluxDB

FROST-Server	Sensor data (IoT) according to the OGC SensorThings API standard	PostGIS
Dataset Catalogue, Model Catalogue, and Data sources Catalogue (Idra)	General platform data	MySQL
Geo Network	Geospatial metadata	PostgreSQL, PostGIS
VIE-AI	Data for Artificial Intelligence dashboards	Not specified
Data Visualisations and Dashboards (Apache Superset)	Data for Business Intelligence and data visualization	PostgreSQL
Workflow Management and Execution (Apache Airflow)	Data for workflow orchestration (DAGs)	PostgreSQL
Security and User Management (Keycloak)	Data for user, identity, and access management (IAM)	PostgreSQL
NBS Registry	Data related to registries (e.g., Nature-Based Solutions)	MongoDB

4.1.3 Object Storage

4.1.3.1 Description and Purpose

The Object Storage provides the ability to handle large amounts of data files, both structured data formats (e.g., large CSV datasets) and unstructured data like multimedia files (e.g., images, videos, etc.). Each stored file (referred to as an object) is uniquely identified and is backed by metadata that describes its properties. The architecture provides for quick access, metadata augmentation, versioning, and data life cycle management. Object storage is responsible in URBREATH for ensuring stable access to raw and processed data for other components.

4.1.3.2 Key Features and Functionality

The Object Storage plays a key function in supporting data analysis and simulation activities through enabling access to large datasets and providing an assured place to store the results generated by the processes. Its key functionalities are:

- Scalable Object Repository: stores data in immutable objects with mass file ingest and retrieval.
- S3-Compatible APIs: offers a RESTful API for programmatic access via standard protocols, offering wide client compatibility.
- Metadata Management: stores all objects with user-custom metadata to enable tagging, categorization, and semantic annotation.
- Versioning and Lifecycle Policies: offers object versioning and automated lifecycle policies for archiving, retention, or deletion.

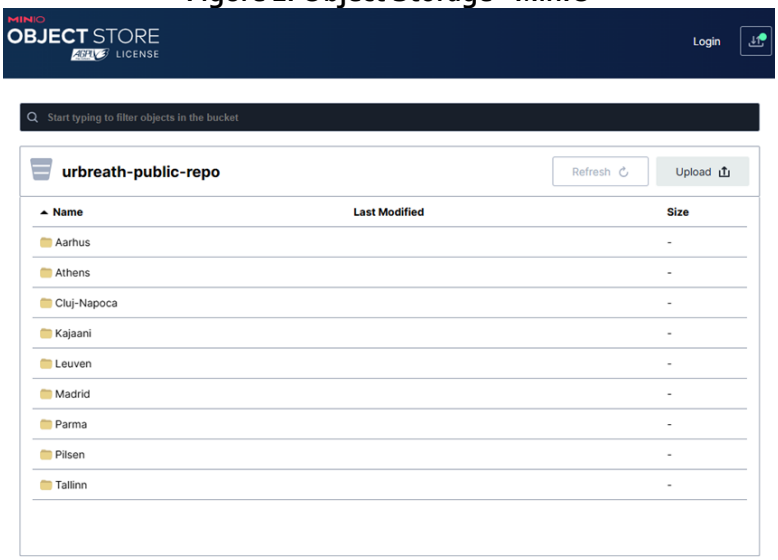
- Security and Access Control: offers rich permissions, encryption, and bucket policies to support secure data isolation and sharing.
- Fault Tolerance and Replication: offers data availability and durability through the replication of nodes or data centers.

4.1.3.3 Technical implementation

Object storage in URBREATH is achieved using **MinIO**, a high-performance object storage system that is S3-compatible and intended for cloud-native deployments. MinIO is executed as a containerized application and configured to offer S3-compatible APIs such that internal components can use it the same way they would AWS S3.

One of the key aspects of the deployment is the bucket notification system: URBREATH configures buckets to notify a single or multiple external endpoints with events (such as object creation or deletion). This enables URBREATH toolbox to realize reactive behavior, catalogues updatability, triggering downstream data processing, or publishing metadata entries on new data availability.

Figure 2: Object Storage - MinIO



In terms of security, MinIO enforces strict authentication and access control through a combination of internal and external mechanisms:

- All clients need to be authenticated using the AWS Signature Version 4 protocol. Any request needs to support a valid access key and secret key.
- MinIO includes an internal Identity Provider (IDP) for user and credential management.
- It also supports external identity federation through OpenID Connect (OIDC) providers, Active Directory, or LDAP services - allowing seamless integration with organizational IAM infrastructures.
- Authorization is managed by policy-based access control, which determines what actions (read, write, list, delete, etc.) are allowed per user, bucket, or object.

For protecting sensitive data, MinIO provides end-to-end encryption:

- In-transit encryption ensures that data in transit over the network is encrypted via TLS.
- At-rest encryption provides the facility that the stored objects get encrypted in buckets such that even during storage compromise, the risk of exposure of data gets lowered down to zero.

Multi-tenant access control is provided by MinIO, under which different URBREATH services and users can operate under disjoint namespaces. Metadata tags can be assigned to all objects stored for describing content type, source, date of creation, and geographic context, which can be used by the indexing or cataloguing components. Versioning and access control policies are configurable via MinIO's admin interface or S3-compatible SDKs, enabling flexible and secure data management.

4.1.4 GeoServer

4.1.4.1 Description and Purpose

GeoServer is an open-source server application that enables users to share, process, and edit geospatial data. It supports a wide range of data formats and standards, making it a key component in spatial data infrastructures.

4.1.4.2 Key Features and Functionality

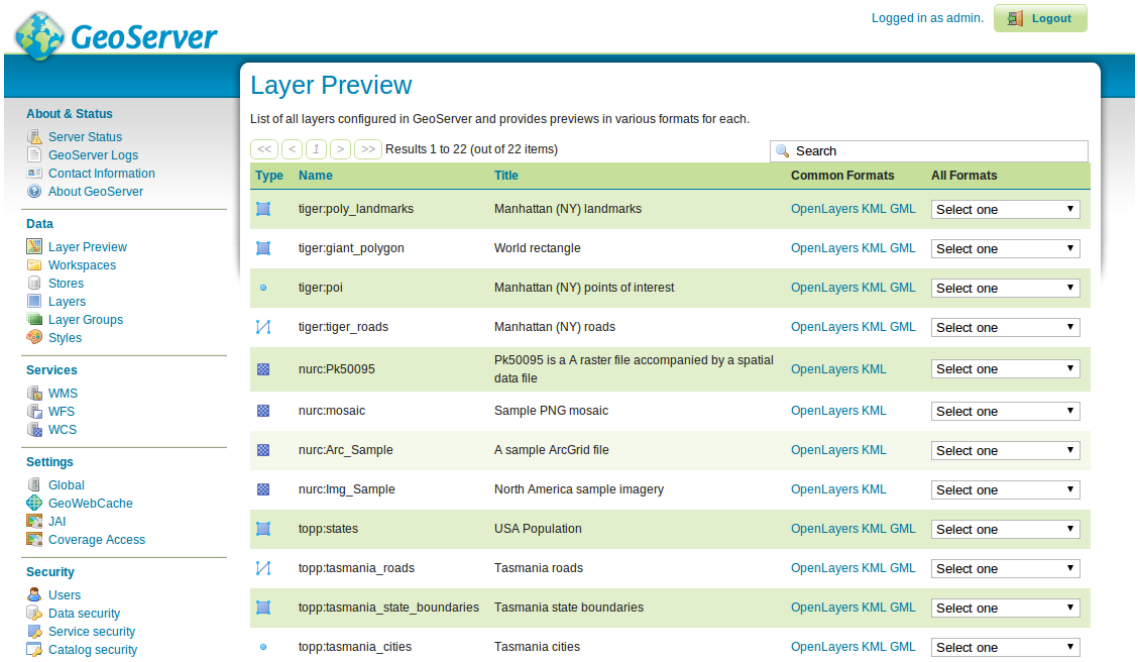
Using open standards set forth by the Open Geospatial Consortium (OGC), GeoServer allows for flexibility in map creation and data sharing.

Designed for interoperability, it publishes data from any major spatial data source using open standards.

GeoServer can also display data on any of the popular mapping applications such as Google Maps, Google Earth, Microsoft Bing Maps, and MapBox.

In addition, GeoServer can connect with traditional GIS architectures such as ESRI ArcGIS or QGIS.

Figure 3: GeoServer[13]



4.1.4.3 Technical implementation

GeoServer is developed in Java and runs on any platform that supports a Java Runtime Environment (JRE). It is a web application that can be deployed as a standalone server or within a Java Servlet container such as Apache Tomcat.

GeoServer requires a Java 11 or Java 17 environment (JRE) to be installed on the system; This must be done prior to installation.

4.2 Data Connectors

4.2.1 Data Catalogues Connectors

The Data Catalogues Connectors allow the interconnection between the Dataset Catalogue of the URBREATH Toolbox and catalogue repositories external to the URBREATH Toolbox. These connectors represent the connection point that allows the Dataset Catalogue to "federate" heterogeneous catalogues within a homogeneous catalogue accessible to the end user via a Web UI and REST APIs.

[13] Source: https://live.osgeo.org/it/quickstart/geoserver_quickstart.html

For this purpose, Data Catalogues Connectors support different technologies and software systems employed to build catalogues of datasets, such as CKAN and SOCRATA, giving the chance to interact with the APIs they expose, collect the metadata about datasets they manage, and uniform the representation of these metadata according to DCAT-AP.

Table 3: Data Catalogues Connectors[14]

Connector	Supported version	Description
CKAN	API v3	Connector for CKAN portals
DCATDUMP	DCAT_AP v1.1, DCAT-AP_IT v1.0	Connector to import metadata via dump file compliant with DCAT-AP or DCAT-AP_IT
DKAN	DCAT Catalog endpoint	Connector for DKAN portals
JUNAR	API v2.0	Connector for Junar portals
NATIVE	API v1.0	Connector for Idra Federation API
OPENDATASOFT	Search API v2	Connector for OpenDataSoft portals
ORION	NGSIv2, NGSI-LD	Connector that allows to create datasets with NGSIv2/NGSI-LD query-based distributions
SOCRATA	DCAT Catalog endpoint	Connector for Socrata portals
SPARQL	v1.1	Connector that allows to create datasets with SPAQRL query-based distributions
SPOD	CKAN API v1&2	Connector for Stat Portal Open Data (SPOD)
WEB	v1.0	Connector to import sitemaps in order to scrape and federate custom websites
Zenodo	V1.0	Connector to import all datasets linked to a community

Data Catalogues Connectors are part of the Dataset Catalogue, and from a technical point of view they are embedded into **Idra**, the technical assed employed for the implementation of the Dataset Catalogue (section 4.8.1)

[14] Source: Idra documentation; https://idra.readthedocs.io/en/latest/architecture/architecture/#architecture-overview

4.2.2 IoT Connectors

4.2.2.1 Description and Purpose

IoT Connectors in the URBREATH architecture are responsible for ensuring transparent communication between a wide range of IoT devices and the rest of the platform. The connectors facilitate the real-time sensor data collection by supporting different communication protocols commonly used in the IoT ecosystem (e.g., MQTT, LoRaWAN, HTTP). Through translating protocol-dependent messages to a common data model, IoT Connectors render the data produced by physical devices consumable in the URBREATH Toolbox.

4.2.2.2 Key Features and Functionality

IoT Connectors within the URBREATH platform are designed to handle different communication protocols handled by IoT devices, such as MQTT, HTTP, LoRaWAN, and Lightweight M2M. This support for multiple protocols ensures that different devices supplied by various vendors with their own communications standards can be integrated without changes to the URBREATH architecture. One of the functionalities asked for in such connectors is that they will be able to translate protocol-specific payloads into standardized data formats - most notably NGSI-LD or the OGC SensorThings API. This translation standardizes all incoming data so that it will be understood and processable by the system regardless of the device origin. Due to this architecture, IoT Connectors make it easy to import live streams of data related to environment, mobility, and urban infrastructure into the context layer of the platform. They also facilitate periodic device management activities, such as registering new sensors, monitoring their functioning, or updating configurations, in protocol-agnostic interfaces.

4.2.2.3 Technical implementation

In URBREATH, IoT data integration is performed using IoT Agents, middle-tier elements that are protocol translators between the devices and other components such as Frost Server and Orion-LD Context Broker. One IoT Agent controls a specific communication protocol and translates the messages of the device into NGSI-LD context information, OGC SensorThings API. IoT Agents act as a bridge, allowing devices to send information using their native protocols without blocking the URBREATH platform from obtaining semantically enriched interoperable context data. The agents are modular, extensible, and comply with the FIWARE open-source standard. They also support interaction with external identity providers and policy-based access control according to the security requirements of the platform. Below there is a summary of the most used IoT Agents and supported protocols.

Table 4: IoT Agents

IoT Agent	Protocol	Description
IoTAgent-JSON	MQTT / HTTP	Intercepts MQTT/HTTP messages with JSON payload to map into NGSI context data
IoTAgent-LWM2M	Lightweight M2M	Supports LWM2M protocol, bridging it with NGSI for device management and telemetry
IoTAgent-UL	UltraLight 2.0	Handles UltraLight 2.0 payloads over MQTT/HTTP and maps them to NGSI entities

IoTAgent-LoRaWAN	LoRaWAN	Converts messages from LoRaWAN networks into NGSI format for integration into the context broker
IoTAgent-Sigfox	Sigfox	Integrates Sigfox IoT messages with the FIWARE context model
IoTAgent-ISOXML	ISOXML	Translates ISOXML-based telemetry (used in precision agriculture) into NGSI-compliant context data

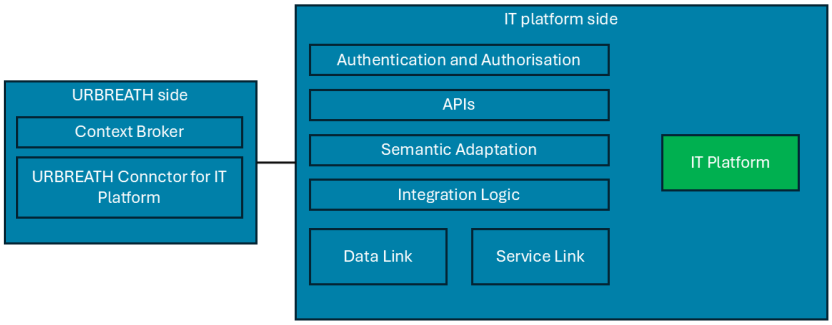
4.2.3 IT Platform Connectors

4.2.3.1 Description and Purpose

IT Platform connector is a set of functional blocks designed to enable secure, standard and interoperable communication between the Toolbox and external IT systems used in the data environments. These connectors facilitate integration of other systems such as legacy ones, municipal databases, and third-party services by providing a modular interface to handle data exchange. It's important to note that this connector is not a dedicated software asset, but it provides guidelines for specific implementations of this kind of connector, since each implementation is tailored for a specific IT system of a Municipality to be interconnected to the URBREATH Toolbox.

The connector is conceived to operate as a bridge, allowing the URBREATH Toolbox to both retrieve data from and deliver data to external IT platforms without disrupting existing infrastructure. It supports URBREATH's overall aim of promoting federated, cross-domain urban analytics, while respecting local data governance, security policies, and technical constraints.

Figure 4: IT Platform Connector



4.2.3.2 Key Features and Functionality

The URBREATH Toolbox Connector incorporates a set of functional blocks, each responsible for a specific aspect of the integration process:

Table 5: IT Connector functional blocks

Authentication and Authorization	Enforces secure access to data and services, protects endpoints from unauthorized or malicious access, enforces the policies defined by both the platform and the external system

API	Provides a standardized interface through RESTful APIs for interacting with the connected system. These APIs ensure a uniform method for data retrieval, submission, and service invocation, customized to the technical specifics of each IT platform
Semantic Adaptation	Translates exchanged data formats into the semantic models used by the system. This ensures that incoming and outgoing information is semantically aligned with the URBREATH knowledge framework, supporting data interoperability
Integration Logic	Manages coordination and execution of integration workflows, including adaptation to the legacy system's constraints, formats, and security requirements. Handles tasks such as data transformation, anonymization, and orchestration of access protocols
Data Link	Facilitates direct access to the data managed by the connected IT Platform. Based on the platform's specific data sharing policies, technical limitations, and data governance rules
Service Link	Enables interaction with the services provided by the connected platform. This allows to not only access data but also trigger or manage operations remotely

4.2.3.3 Technical implementation

The connector architecture is split between URBREATH and external IT Platform side. The technical setup involves deploying a set of standardized REST endpoints that manage different aspects of data exchange:

/proxy: Accepts incoming data requests forwarded by authorized entities
/data: Receives data payloads from sending systems
/about/version: Returns metadata about the connector's configuration and capabilities

These endpoints follow protocols based on HTTP and HTTPS, depending on the policies. The connector's API specification ensures compatibility with existing data systems and is informed by the TRUE Connector and IDS architectural patterns, promoting openness and standardization. To adapt to diverse urban IT environments, the Integration Logic Layer is designed to be extensible and configurable. It supports variable authentication schemes, multiple data formats, and a range of security protocols.

The URBREATH Toolbox Connector for IT Platforms provides a secure and flexible solution for integrating different urban systems, adopting a layered and modular approach to ensure interoperability. This enables the URBREATH Toolbox to deliver on its goal of facilitating rich, cross-domain urban analytics while maintaining compatibility with existing city infrastructures.

For the implementation of the functional blocks, some tools already available in the URBREATH Toolbox can be employed.

Concerning the Authentication and Authorisation capabilities, the recommended tool is Keycloak.

Semantic adaptation can be supported by Apache Airflow (i.e. Workflow Management and Execution). For the implementation of the Integration Logic, the Data Link and the Service Link Apache Airflow can also be employed; however, since they are closely related to the technical characteristics of the IT Platform to be interconnected with the URBREATH Toolbox (e.g. Exposed APIs, databases, framework, programming languages, etc.), in many cases, it could be necessary to implement them from scratch.

4.2.4 Data Repositories Connectors

4.2.4.1 Description and Purpose

URBREATH Toolbox aims at enabling integrated urban analytics by bringing together data from multiple sources relying on a broad mix of data systems. To access these sources efficiently, the Toolbox uses **Presto DB**, an open-source, distributed SQL query engine designed for fast federated data access. With Presto's connectors, URBREATH can query across disparate systems without need to move or duplicating data, providing a unified, flexible interface for cross-domain urban data analysis, minimizing infrastructure complexity.

4.2.4.2 Key Features and Functionality

The Presto-based data connectors offer several features to align with the Toolbox's requirements for scalability, interoperability, and data sovereignty. Main ones can be described as follows:

Table 6: Data Repository Connectors

Federated Querying	Users can query multiple data sources simultaneously using standard SQL, without needing to centralize data
Support for Heterogeneous Sources	Connectors are available for a wide range of technologies, including PostgreSQL, MySQL, SQL Server, MongoDB, Redis, Amazon S3, Apache Kafka, and Hadoop-based systems
On-Demand Data Access	Data remains in place, allowing URBREATH to work with real-time, historical, and static data without prior transformation or replication
Extensibility	Additional connectors can be configured to integrate new systems or data repositories
Transparency and Monitoring	Availability of a web interface which allows users to monitor queries and system performance in real-time, supporting operational oversight

4.2.4.3 Technical implementation

Within the architecture, Presto is deployed as a distributed system composed of a Coordinator node and multiple Worker nodes with different purposes. Coordinator handles query parsing, planning, and management, while the Workers are responsible for executing query tasks and retrieving data from connected sources.

Data repositories are connected to **Presto** through a dedicated connector module configured through simple properties files. These configurations define access parameters such as endpoints, authentication, and metadata mappings. Once a connector is active, its datasets appear within Presto's unified catalogue, allowing them to be queried like other SQL tables.

4.2.5 Data Spaces Connector

4.2.5.1 Description and Purpose

The Data Spaces Connector is a key part of the URBREATH setup. It helps to let safe, uniform, and policy-driven data sharing happen between groups in a joined and trusted zone - called a Data Space. It fits with the rules of the International Data Spaces (IDS) plan, and the connector makes sure of data control, use checks, and working together well. Within URBREATH, it aids in teamwork across areas and borders with other setups and smart city spaces, supporting trusted and clear data sharing.

4.2.5.2 Key Features and Functionality

A Data Spaces Connector is designed to support these capabilities:

- Trusted Data Exchange: it makes sure that talks between data givers and users are safe and can be checked. This keeps data true and real.
- Usage Control Enforcement: it sets clear rules on who can use the data, how they can use it, and what it's for.
- Role Flexibility: it can act as both the one giving data (letting others reach certain data sets) and the one getting data (asking for data from other places).
- Interoperability: it follows world rules like the IDS Reference Model and works well with other IDS-ready connectors and help.
- Protocol and Format Flexibility: it uses many ways to send data (like HTTPS, IDSCPv2, WebSocket) and many types of messages (like multipart/mixed, JSON-LD, HTTP headers).
- Policy-Aware Interaction: Each transaction is evaluated based on predefined contracts and policies, ensuring alignment with data governance requirements.

4.2.5.3 Technical implementation

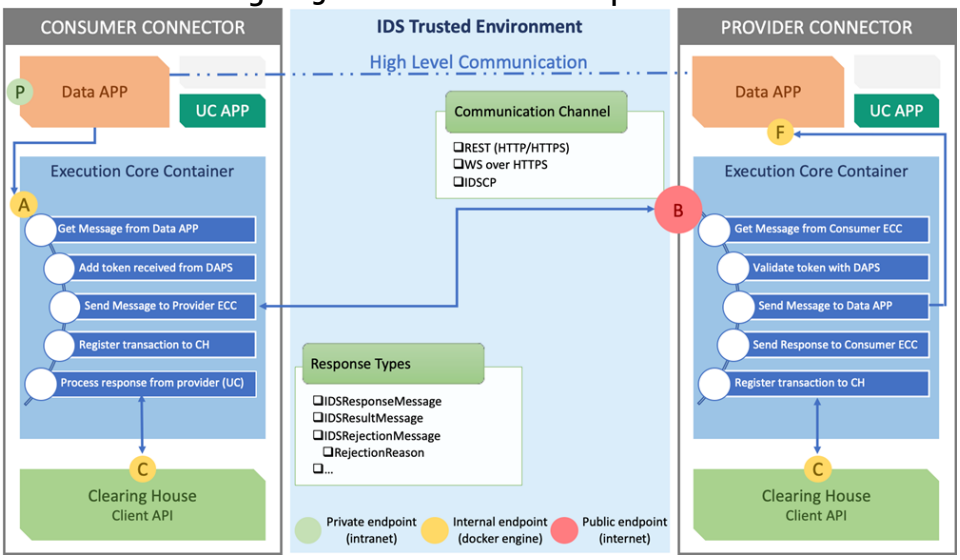
In URBREATH, the **Data Spaces Connector** is implemented using the FIWARE TRUE Connector [15] (TRUsted Engineering Connector), a modular and open-source solution which is IDS-compliant. Under the AGPL v3 license, the TRUE Connector has been designed to be very configurable and integratable into a variety of varied smart city use cases.

15 https://fiware-true-connector.readthedocs.io/en/latest/

The TRUE Connector is composed of three main components:

- Execution Core Container (ECC): execution core engine responsible for processing secure data exchange operations.
- FIWARE Data Application: processes incoming data requests and passes on the resultant responses according to business logic.
- Usage-Control Data Application: verifies whether the requesting entity is entitled to receive the data, according to usage policies specified.

Figure 5: TRUE Connector components[16]



The TRUE Connector can operate in consumer mode, requesting data access to external providers, and provider mode, providing data to other connectors based on access control decisions. It can transport data through various transport protocols, for example, HTTP, HTTPS, WebSocket over HTTPS, and IDSCPv2, and can deal with a range of content types such as multipart/mixed or multipart/form.
In URBREATH, the TRUE Connector acts as the gateway for secure data sharing to ensure that all outgoing and incoming data exchange adhere to the prescribed usage control rules and bring the platform closer to larger objectives like the Common European Data Spaces.

4.3 Components Communication

4.3.1 Message Broker

4.3.1.1 Description and Purpose

The message broker component of the URBREATH architecture is intended to facilitate the reliable, real-time exchange of data between heterogeneous systems, such as IoT devices, data processing

[16] Source: https://fiware-true-connector.readthedocs.io

services, analytical engines, and dashboards. As a communication hub, the message broker implements a publish/subscribe model, in which producers send data to named channels (topics), while consumers receive the data either in real time or in deferred mode.

Its primary goals are to:

- Decouple producers and consumers, allowing systems to operate independently and asynchronously,
- Offer scalability and fault tolerance in data exchange workflows,
- Allow streaming-based applications such as environmental monitoring, data ingestion pipelines, microservice orchestration, and real-time analytics.

This architectural layer ensures URBREATH's data flows are robust, consistent, and adaptable to varying system loads and integration scenarios.

4.3.1.2 Key Features and Functionality

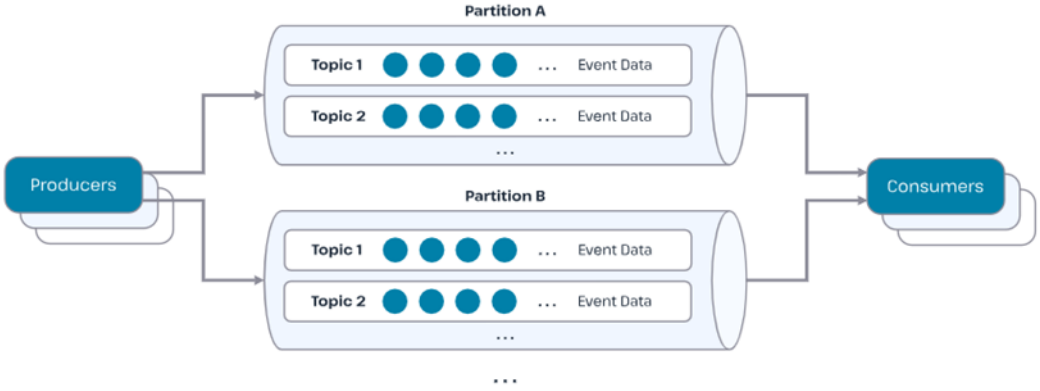
The message broker in URBREATH provides a rich set of features for high-performance data streaming and integration:

- Distributed Messaging & Event Streaming: messages are distributed across multiple nodes and replicated for high availability and fault tolerance.
- Topic-based Publish/Subscribe: data is sent via topics, which can be partitioned for parallel and scalable consumption.
- Durable Storage with Retention Policies: persistently stores the messages that can be replayed or stored based on configurable retention policies.
- High Throughput & Low Latency: supports millions of messages per second with sub-second latency, suitable for real-time processing.
- Consumer Group Support: provide load-balanced and coordinated consumption by multiple consumers, with offset tracking to ensure reliability.
- Stream Processing Integration: supports integration with real-time data processing frameworks to enable complex event processing.
- Connector Ecosystem: provides extensibility for integrating with external systems (e.g., databases, APIs, file storage).
- Security & Access Control: secures communication using encryption and access control mechanisms.
- Monitoring & Observability exposes performance metrics and system health indicators for operational visibility.

4.3.1.3 Technical implementation

In URBREATH, the message broker is implemented using Apache Kafka, a distributed, high-performance event streaming platform. Kafka provides the messaging backbone within the centre, managing the flow of real-time data between internal components through a publish/subscribe model.

Figure 6: Message Broker (Apache Kafka)[17]



Producers (including IoT gateways and internal URBREATH services) publish data to the topics. Consumers process the data either in real time (for dashboards and alerts) or batch mode (for analytics and reporting). Kafka's commit log model ensures durable storage and replicability of all messages.

Kafka provides native capabilities to connect to external systems for fault-tolerant and scalable data transfer from and to external data stores such as relational databases, NoSQL databases, and cloud services. The integrations allow for seamless creation of end-to-end data pipelines without custom code.

Kafka also enables native stream processing capabilities, enabling real-time data transformation and analysis as it travels through the system. This includes native support for complex operations such as filtering, aggregations, joins, and time-based windowing, all executed within the Kafka stack without the need for external processing engines.

To ensure structured and consistent data exchange, schema management features allow messages to conform to predefined formats. These capabilities enforce validation during serialization, maintain compatibility across message versions, and help prevent errors during data processing.

4.3.2 Context Broker

4.3.2.1 Description and Purpose

The URBREATHER platform's context broker saves, organizes, and publishes the entities' and systems' state in real time. It provides a global, queryable layer of knowledge where all entities are semantically defined and continuously updated based on real-world modifications.

The context broker represents a digital twin of the city, where each entity (e.g., a sensor or traffic signal) is represented with attributes (e.g., temperature, CO_2 level) and relationships (e.g., locatedIn, connectedTo). Its role is to ensure semantic consistency, interoperability, and reliability in accessing

[17] Source: https://hazelcast.com/foundations/data-and-middleware-technologies/kafka/

and updating live urban data, serving as the foundation for real-time decision-making, automation, and smart city intelligence.

4.3.2.2 Key Features and Functionality

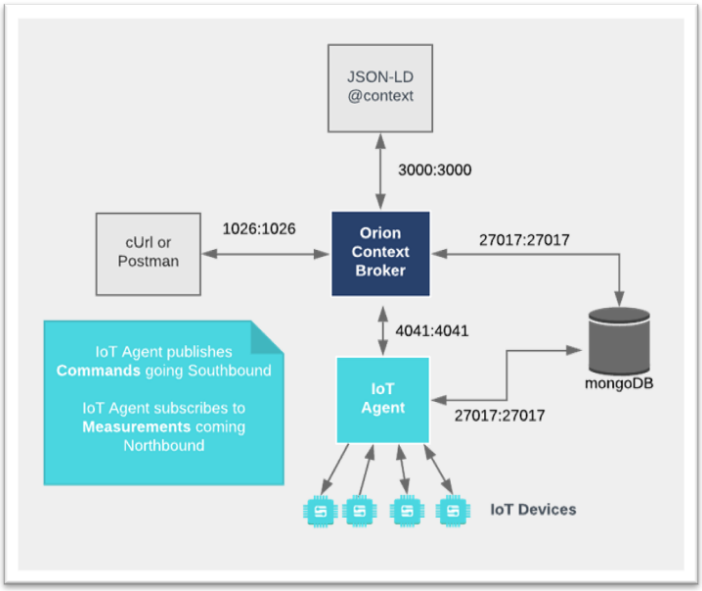
The context broker in a smart city architecture offers the following core capabilities:

- Entity Lifecycle Management: creation, update, deletion, and querying of entities and their attributes and relationships.
- Standardized Data Models: semantically enriched data support with formalized data formats like JSON-LD for machine-readable context.
- Subscription & Notification System: Mechanisms to subscribe to context changes and asynchronous receive notifications for some conditions or thresholds.
- Temporal Data Support: Integration with time-series storage components to enable historical queries and trend analysis.
- RESTful APIs: Standard HTTP interfaces for accessing and writing context data.
- Semantic Interoperability: adherence ontologies and vocabularies (e.g., SAREF, DCAT-AP), enabling data federation and integration across domains.
- Cross-Domain Integration: interoperability with additional context brokers and data sources, enabling federated sharing of data within city systems or across regions.

4.3.2.3 Technical implementation

In URBREATH, the context broker is implemented using **Orion-LD (**Context Broker**),** an open-source, NGSI-LD-compliant context broker developed within the FIWARE community. Orion-LD is a successor of the original Orion Context Broker since it has taken over the NGSI-LD standard that utilizes JSON-LD and Linked Data principles to provide semantic interoperability and richer entity modelling.

Figure 7: Context Broker (Orion-LD)[18]



Orion-LD is deployed as a containerized microservice, typically from a stable Docker image. It uses MongoDB as its primary storage backend and is also capable of supporting QuantumLeap[19] and CrateDB[20] to enable time-series storage and analysis of historical context data.

NGSI-LD entities in Orion-LD can be used to model urban objects and phenomena (e.g., air quality sensors, traffic sensors), attributes (such as measurement values and timestamps) and relationships (like spatial or functional associations). "*@context*" files are used to enforce semantic structure, ensuring consistent interpretation of data across different domains.

The subscription and notification functionality allows URBREATH services to respond to context changes in real time. For example, it can trigger downstream services or alerts when specific conditions (e.g., pollution thresholds) are met.

Lastly, Orion-LD supports semantic enrichment using domain-specific ontologies like SAREF and standards such as DCAT-AP, enabling interoperability with other NGSI-LD-compliant platforms. This makes URBREATH capable of federating urban data across systems and municipalities, supporting scalable and modular smart city deployments.

---

[18] Source: https://github.com/FIWARE/tutorials.NGSI-v2/blob/master/docs/iot-agent.md

[19] QuantumLeap is a REST service for storing, querying and retrieving NGSI v2 and NGSI-LD spatial-temporal data; https://github.com/orchestracities/ngsi-timeseries-api

[20] CrateDB is a distributed SQL database management system that integrates a fully searchable document-oriented data store. It is open-source, written in Java, based on a shared-nothing architecture, and designed for high scalability. https://github.com/crate/crate

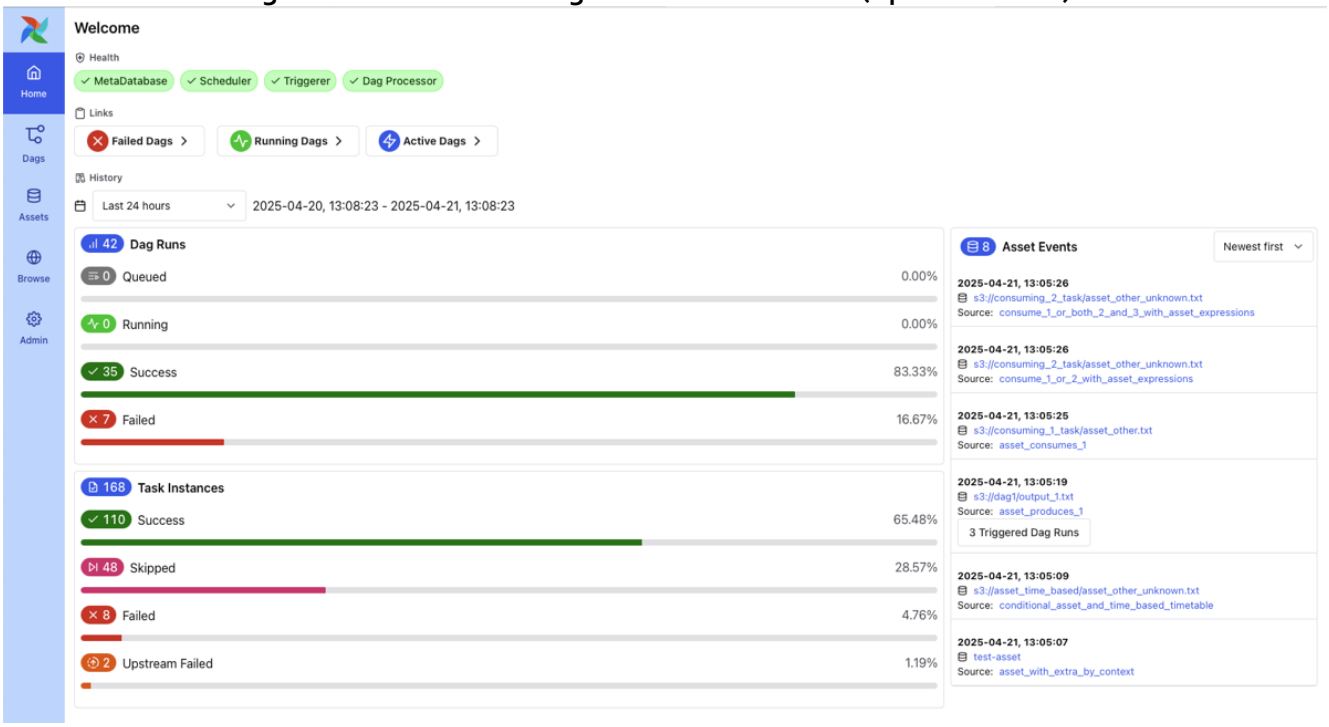
4.4 Process Orchestration

4.4.1 Workflows Management and Execution

4.4.1.1 Description and Purpose

Within Process Orchestration macro area, Workflow Management and Execution is a fundamental element that supports the definition, management, and execution of processes. It also coordinates interactions among different tools of the URBREATH Toolbox to achieve results efficiently. Apache Airflow has been selected as the tool for implementing this functionality, given its robust ecosystem. Airflow uses Python scripts for workflow definitions, enabling flexible and programmatic orchestration of pipelines, combined with scheduling and process monitoring features accessible via a user-friendly web-based UI.

Figure 8: Workflows Management and Execution (Apache Airflow)[21]



[21] Source: https://github.com/FIWARE/tutorials.NGSI-v2/blob/master/docs/iot-agent.md

4.4.1.2 Key Features and Functionality

The main features of this component can be summarized as follows in Table 7.

Table 7: Workflows Management and Execution – Main features

Workflow Definition via Python Scripts	Workflows are defined using Python, providing high flexibility in designing process logic and dependencies
Scheduling	Supports sophisticated scheduling options and event-driven triggers to control when workflows execute
Process Coordination	Coordinates the interactions between multiple tools and components within the URBREATH Toolbox to deliver end-to-end process automation
Monitoring and Administration UI	A rich, web-based interface allows users to monitor workflow status, inspect logs, view execution history, and manage workflow configurations
Robust Error Handling and Retry Policies	Tasks within workflows have configurable retries and failure management, ensuring resilience and minimizing downtime
Parallelism and Scalability	Supports parallel execution of tasks and can scale horizontally by distributing workloads across multiple nodes
Integration Capabilities	Seamlessly connects with various data sources, processing frameworks, and external services within the Toolbox ecosystem

4.4.1.3 Technical implementation

The Workflow Management and Execution functionality is implemented through Apache Airflow, following these key technical details:

- DAG Based Workflow Definition: Each workflow is encapsulated in an Airflow Directed Acyclic Graph (DAG) defined as a Python script. The DAG outlines tasks, their dependencies, schedules, and execution parameters.
- Task Operators: Workflows leverage a mix of built-in Airflow operators and custom operators designed to interact with various URBREATH Toolbox components.
- Dependency Management: Task dependencies are specified programmatically using Airflow's intuitive syntax, ensuring tasks execute in the correct sequence.
- Scheduler and Executor: Airflow's scheduler manages the timing and triggering of workflows according to configured schedules and external events.
- Process Monitoring and Logging: The Airflow UI provides detailed real-time monitoring of workflow executions, including task status, logs, and historical performance metrics.
- Alerting and Notification: Integrated alerting mechanisms notify operators of task failures, retries, or successful completions via e-mail or messaging platforms such as Slack.

This implementation ensures that complex, multi-component processes within the URBREATH Toolbox are orchestrated reliably and transparently.

4.5 Data Analysis and Processing

4.5.1 Heat Stress Analysis

4.5.1.1 Description and Purpose

This module is designed to assess urban heat Islands using satellite-derived Land Surface Temperature (LST) data. It highlights critical areas within urban environments that are significantly warmer than surrounding rural areas due to human activities such as urbanization, infrastructure density, and vegetation loss. The primary output is a percentage-based Urban Heat Island (UHI) Intensity Index (0–100). Use cases for this index can be found in sectors such as climate adaptation planning, public health interventions, and energy demand forecasting.

4.5.1.2 Key Features and Functionality

The Urban Heat Island (UHI) analysis pipeline integrates multiple satellite data sources (including L40 LST at 10 meters, Landsat-8/9 at 30 meters, and MODIS at 1-kilometer resolution) to support a wide range of spatial analyses. The temporal resolution of this tool can be every 5 to 7 days, or daily. Users can select between monthly, seasonal, or annual temporal aggregations, depending on their monitoring needs. The workflow is fully automated, though it allows the optional input of a custom rural reference shapefile for increased precision. The output is a GeoTIFF raster with normalized UHI index values, ranging from 0 to 100, expressed as percentages. Built for scalability, the pipeline supports global application, with automated rural area detection ensuring adaptability across diverse landscapes. Impressively, the system can process up to 100 square kilometres in approximately 20 minutes on standard infrastructure, offering both speed and reliability.

4.5.1.3 Technical implementation

Urban Heat Islands (UHIs) are computed by measuring the difference between the Land Surface Temperature (LST) of urban areas and that of a rural reference area.

The process begins with the user defining the Area of Interest (AOI) and optionally providing a custom shapefile for the rural baseline. If not provided, an automatic methodology identifies rural areas based on land cover (LC) classification and tree cover density (TCD). The LC map filters out non-natural land types, focusing on greenery and agricultural classes, while the algorithm ranks candidate areas by high tree canopy cover and low population density—key indicators of rural characteristics such as sparse infrastructure and low anthropogenic heat emissions.

LST data, downscaled to 10 m resolution using Sentinel-2 and Landsat-8/9 imagery, is then retrieved and pre-processed, including cloud masking and temporal filtering. Median LST is computed pixel-wise over the selected time.

For each urban pixel, UHI intensity is calculated as the difference between its LST and the median LST of the identified rural area:

$$UHI\ Intensity\ pixel = LST\ pixel - LST\ rural$$

This intensity is then normalized to a 0–100 scale using:

*UHI pixel= (UHI Intensity pixel - min (UHI Intensity) / max (UHI Intensity) - min (UHI Intensity)) * 100*

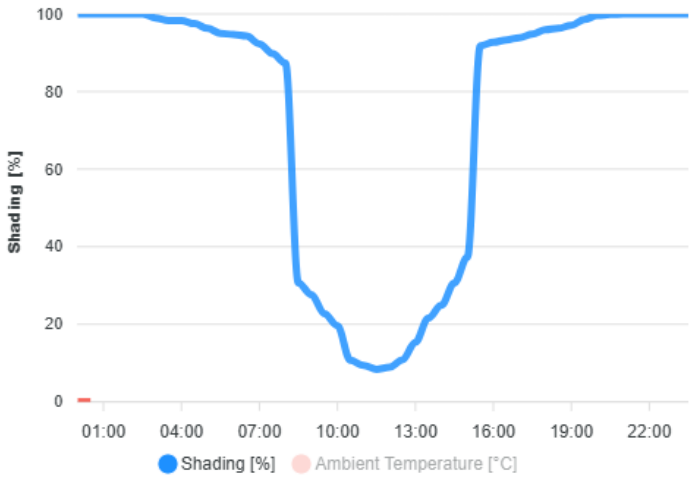
The final output includes a Geo TIFF raster of the UHI index, with the option to generate hotspot visualizations and overlays to support mitigation planning.

4.5.2 Shadow Modelling

4.5.2.1 Description and Purpose

The shadow tool is designed to analyse and visualize shadow coverage for a given area and time using CesiumJS and the VC Map framework. In summary, the shadow tool automates the process of calculating, visualizing, and exporting shadow coverage and its effects for a selected area and date, using Cesium JS shadow mapping capabilities and a user-friendly interface.

Table 8: Shadow Modelling Results



4.5.2.2 Key Features and Functionality

Its main features and workflow are:

1. Shadow Analysis

The tool calculates the shadow ratio (the proportion of an area covered by shadow) for a user-defined polygon on the map.

It does this by:

- Projecting the polygon coordinates to the CesiumJS scene.
- Rendering the scene with shadows enabled for specific time intervals throughout a day.

- Reading the shadow map (a WebGL framebuffer) to determine which pixels within the polygon are in shadow.

2. Time Series

The tool iterates over a full day (in 30-minute steps), updating the sun position and recalculating the shadow ratio for each time step.

For each time step, it stores:

- The timestamp.
- The calculated shadow ratio.
- A PNG image (base64) of the shadow map for that time.

3. Visualization & Export

The results are visualized in a chart (using ApexCharts[22]) showing shadow ratio and estimated temperature over time.

The user can export the results as a PDF, which includes:

- The chart.
- The shadow map images for each time step.
- Tabular data of shadow ratios and temperatures.
- Project and date information.

4. Customization

The tool allows for customization of material properties (e.g., concrete/glass share, absorption) and environmental parameters (e.g., temperature, radiation).

These parameters are used to estimate temperature changes due to shadowing.

5. Integration

The tool is integrated into a Vue.js UI, providing dialogs for export and user input.

It interacts with the VC Map app instance to control the map, layers, and CesiumJS scene.

[22] An open-source JavaScript library to build interactive data visualizations; https://github.com/apexcharts/apexcharts.js

4.5.2.3 Technical implementation

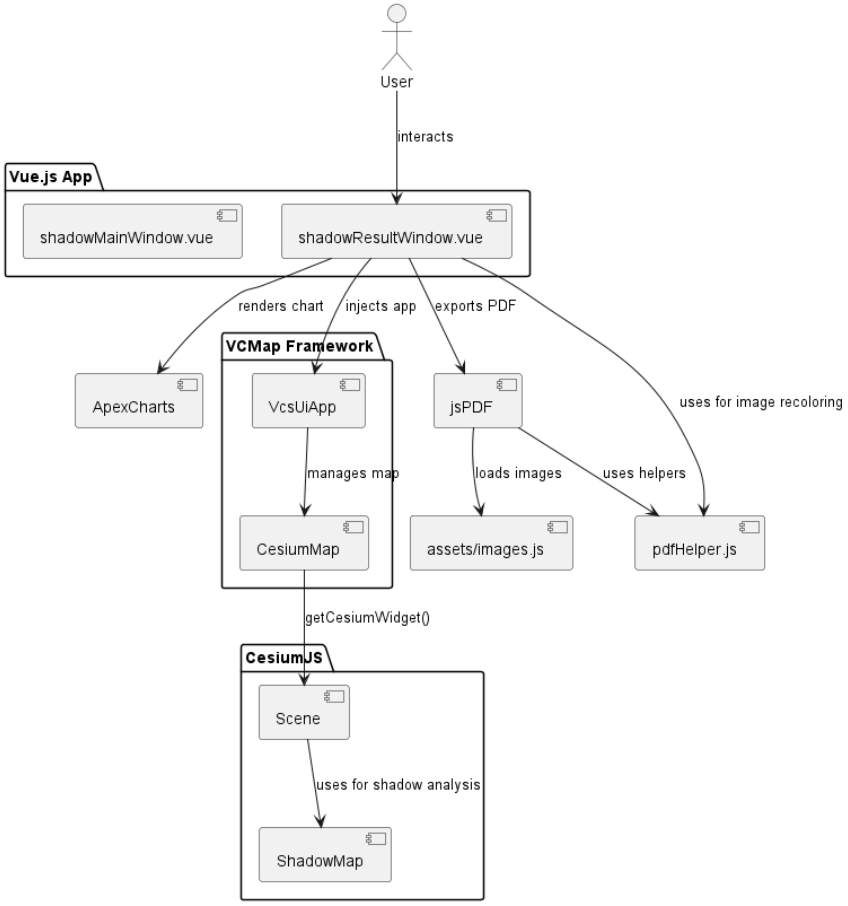
The shadow analysis tool leverages CesiumJS and the VCMap framework to compute and visualize shadow coverage over a user-defined area and time. The process involves the following steps:

1. Scene Setup: The tool configures the Cesium scene, including activating shadows and adjusting the sun position for each time step using the JulianDate API[23].
2. ShadowMap Extraction: For each time interval, the tool reads the shadow map directly from the WebGL framebuffer using gl.readPixels. This provides pixel-level data indicating shadowed and non-shadowed areas.
3. Polygon Projection: The user-defined polygon is projected from map coordinates to screen coordinates using Cesium's SceneTransforms.worldToWindowCoordinates.
4. Pixel Analysis: The tool iterates over the pixels within the polygon's bounding box, using a point-in-polygon test to determine which pixels are inside. It then compares the shadow map data to a base image to calculate the shadow ratio.
5. Result Aggregation: For each time step, the tool stores the shadow ratio, timestamp, and a PNG image of the shadow map.
6. Visualization & Export: Results are visualized using ApexCharts and can be exported as a PDF report with jsPDF, including charts, images, and tabular data.

This approach enables efficient, high-resolution shadow analysis and reporting for urban planning and environmental studies.

[23] https://cesium.com/learn/ion-sdk/ref-doc/JulianDate.html

Figure 9: Shadow modelling - implementation components



4.5.3 Adaptive Rainfall-Infiltration Tracking

4.5.3.1 Description and Purpose

The Adaptive Rainfall-Infiltration Tracking model, developed within the URBREATH project, is a diagnostic and predictive tool aimed at assessing water infiltration in urban environments, especially under changing climate and land use conditions. Initially tailored for Leuven, the model integrates environmental and hydrological data—such as soil moisture, infiltration rates, land cover, and precipitation—to simulate how water moves through urban soils.

The model's purpose is to support urban planners and environmental managers by enabling scenario-based analysis of land cover changes and their impact on stormwater infiltration. Although development is currently focused on Leuven, partner cities like Cluj-Napoca, Kajaani, and Aarhus have expressed interest in adopting it.

Future iterations will include urban surface types and provide user-friendly outputs like heatmaps, scenario analysis, and statistical assessments, making the tool a key asset in climate adaptation and nature-based planning.

4.5.3.2 Key Features and Functionality

The infiltration prediction model provides comprehensive analysis capabilities for urban water management through a combination of machine learning predictions and physics-based modelling. The tool enables rapid assessment of infiltration performance across different surface types and weather scenarios, supporting evidence-based decision-making for sustainable urban development and nature-based solutions implementation.

Core capabilities

- *Multi-surface prediction* evaluates different surface types from impermeable (e.g., asphalt) to high permeability (e.g., rain gardens, bioswales)
- *Real-time weather integration:* processes temperature, precipitation, humidity, and antecedent soil moisture conditions with seasonal adjustment factors
- *Scenario comparison:* enables side-by-side analysis of conventional vs. nature-based solutions (NBS)
- *Prediction accuracy:* achieves $R^2 > 0.95$ performance on synthetic training data with consistent cross-validation results

Prediction engine

- *Machine Learning core:* random Forest regression model with configurable parameters trained on 10.000+ synthetic data points
- *Physics-based foundation* incorporates established soil science equations and literature-derived parameters
- *Surface-specific adjustments:* each surface type has predefined infiltration multipliers and hydraulic properties
- *Weather effect modelling* applies moisture factor (i.e., antecedent precipitation), temperature factor (i.e., freeze effects), and seasonal vegetation factor
- *Feature encoding:* label encoding for surface types, standardized numerical features for ML processing

Input data processing

- *Soil properties:* organic content, clay content, bulk density, porosity, hydraulic conductivity
- *Weather data:* precipitation, temperature, humidity, 7-day antecedent precipitation
- *Surface type selection:* pre-defined surface categories with associated hydraulic properties
- *Temporal factors:* day of year, month, season for vegetation and weather effects

Output Formats

- *Spatial maps:* grid-based infiltration capacity visualization across study areas
- *Temporal heatmaps:* soil moisture and infiltration rates across depths and time intervals

- *Time series forecasts:* daily infiltration predictions with weather correlations over configurable periods
- *Surface comparison matrices:* performance analysis across all surface types under multiple weather scenarios
- *Statistical visualizations:* distribution analysis, correlation heatmaps, sensitivity plots
- *Data exports:* CSV files with prediction results, PNG/PDF visualizations with timestamps

User Interaction

- *Surface type comparison:* select multiple surface types for side-by-side analysis under same conditions
- *Weather scenario testing:* input custom temperature, precipitation, humidity, and past moisture values
- *Time series analysis:* generate forecasts for specified surface types over chosen time periods (days/weeks)
- *Spatial scenario modelling:* compare different urban development scenarios (conventional vs. NBS implementation)
- *Interactive demonstrations:* built-in demo modes for parameter sensitivity analysis and scenario building

4.5.3.3 Technical implementation

The model is implemented as a modular Python-based system that combines established soil science principles with machine learning techniques. The architecture follows a three-layer approach: data input processing, predictive modelling core, and comprehensive output generation. Built using scikit-learn for machine learning, matplotlib for visualization, and pandas for data management, the system is designed for both standalone analysis and potential integration into larger urban planning workflows. The implementation prioritizes computational efficiency, scientific accuracy, and user accessibility while maintaining the flexibility to incorporate real field data as it becomes available.

Core Technologies:

Machine Learning framework

- Algorithm: Random Forest Regressor (scikit-learn)
- Training data: Synthetic dataset based on literature equations and Belgian climate patterns
- Features: 13 input variables (soil properties + weather + temporal factors)
- Performance: $R^2 > 0.95$, RMSE < 3.0 mm/hr

Data Approach - Current vs. Future

This initial version utilizes a synthetic training dataset generated from established soil science literature and realistic weather patterns. The synthetic approach enables rapid model development and comprehensive coverage of surface types and weather scenarios without the time and cost

constraints of extensive field data collection. While this provides excellent comparative analysis capabilities and reliable relative performance rankings between surface types, the model will be enhanced with real field measurements as they become available from ongoing monitoring efforts in Leuven and partner cities. Future versions will integrate actual infiltration measurements, continuous soil moisture monitoring, and site-specific calibration data to improve absolute accuracy and local validation. This hybrid approach balances immediate functionality with long-term scientific rigor.

Data Management

- Surface database: catalogue of 10 surface types with hydraulic parameters
- Weather integration: real-time data processing for temperature, precipitation, humidity
- Spatial processing: grid-based analysis with configurable resolution
- Temporal handling: daily predictions with seasonal adjustment factors

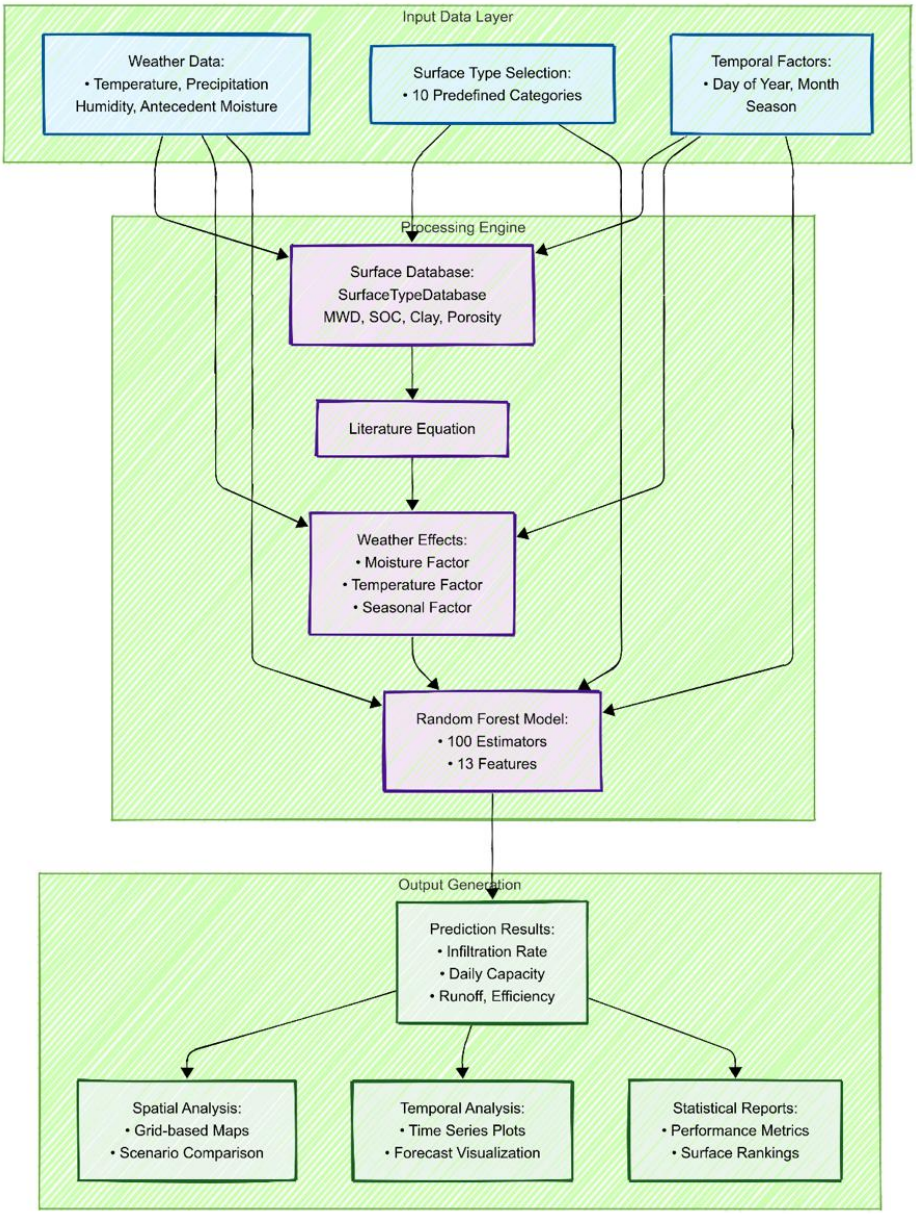
Visualization Engine

- Mapping library: Matplotlib/Seaborn for scientific-grade visualizations
- Interactive plots: multi-panel displays with correlation analysis
- Heatmap generation: soil depth vs. time infiltration tracking
- Statistical charts: distribution analysis, performance comparisons, sensitivity plots

Data Flow(Figure 10**)**

1. Input processing: weather data, surface type selection and soil properties
2. Feature engineering: temporal factors (season, day of year) and moisture calculations
3. ML prediction: Random Forest inference with confidence intervals
4. Output generation: visualization creation, statistical analysis and report formatting

Figure 10: Adaptive Rainfall-Infiltration Tracking - Architecture



4.5.4 Biotope Area Factor (small scale analysis)

4.5.4.1 Description and Purpose

The BAF-Calculation plugin is designed for integration with the VC Map UI application. Its main purpose is to calculate the Biotope Area Factor (BAF) for a given map context. The plugin provides a function (calcBAF) that processes spatial data layers (likely GeoJSON) to compute the BAF, which is a measure of the proportion of ecologically effective surface area in relation to the total land area. The plugin can create a new BAF layer, perform the calculation, and aggregate results such as the total area and the BAF-weighted area. It also integrates with the application's context menu and supports internationalization (i18n) for English and German. In summary, the BAF-Calculation plugin enables users to calculate and analyse the Biotope Area Factor for spatial data within the VC Map UI, supporting ecological planning and assessment workflows.

4.5.4.2 Key Features and Functionality

Calculates the Biotope Area Factor (BAF) for spatial data layers, enabling users to assess the ecological value of land use directly within the VC Map UI.

- Layer Management: Allows users to create and manage BAF-specific map layers, supporting spatial analysis and visualization.
- Result Aggregation: Aggregates and summarizes important metrics such as total area and BAF-weighted area for comprehensive reporting.
- Context Menu Integration: Enhances user interaction by adding custom context menu actions related to BAF calculations on the map.
- Internationalization (i18n): Supports multiple languages (English and German), making the plugin accessible to a wider audience.
- Custom Icons: Provides unique SVG icons for easy identification of BAF-related features and actions in the user interface.
- Plugin Lifecycle Management: Includes robust hooks for initialization, configuration, state management, and cleanup, ensuring smooth integration and operation within the VC Map environment.

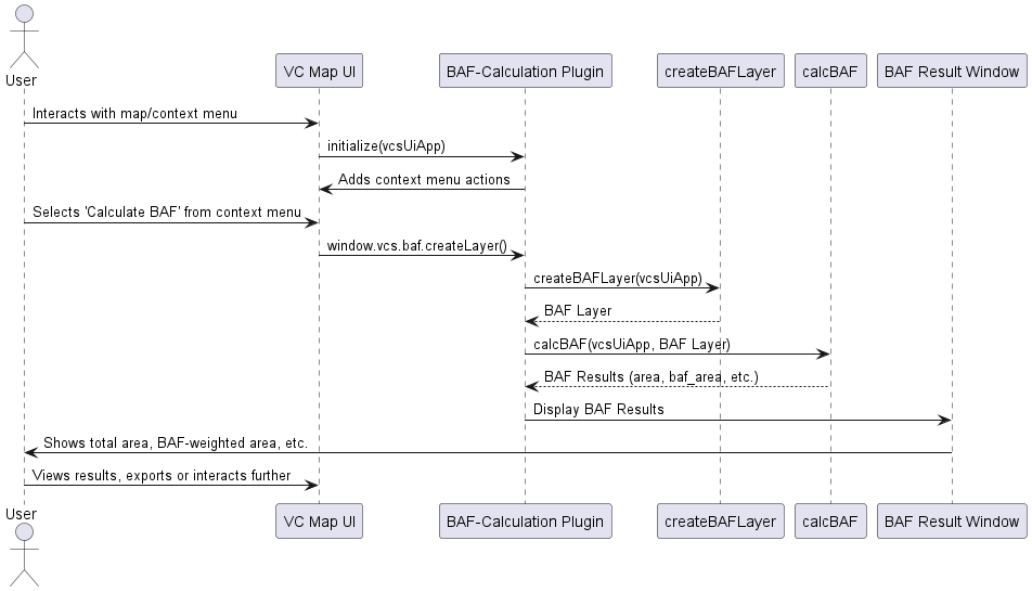
These features make the plugin a powerful tool for ecological planning, spatial analysis, and decision-making in urban and environmental contexts.

4.5.4.3 Technical implementation

The below diagram shows:

- The user initiates BAF calculation via the UI.
- The plugin creates a BAF layer and calculates results.
- Results are displayed to the user, typically in a result window.

Figure 11: Biotope Area Factor (small scale analysis) - Sequence diagram[24]



4.5.5 3-30-300 Analysis

4.5.5.1 Description and Purpose

The 3-30-300 Rule is an urban planning guideline designed to enhance the quality of life in cities by promoting access to nature and green spaces. It consists of three key principles:

- **3 Trees**: every person should be able to see at least 3 trees from their home, supporting visual contact with nature that can improve mental health and well-being.
- **30% Canopy**: neighbourhoods should maintain at least 30% tree canopy cover, ensuring sufficient green coverage that contributes to cooling urban heat islands, improving air quality, and enhancing biodiversity.
- **300 Meters**: every resident should live within 300 meters of a public green space.

4.5.5.2 Key Features and Functionality

The implementation of the 3-30-300 Rule involves three analytical approaches tailored to each principle:

1. ***3 Trees – Visibility Analysis:***

This part assesses the actual visibility of trees from residential buildings by simulating views from "virtual windows" placed on building facades. Tree positions are obtained from various sources, such as low-resolution satellite imagery, detailed tree inventories, or detection of individual tree canopies.

[24] The diagram illustrates user interaction and the output flow of the BAF-Calculation plugin.

The visibility is then calculated considering obstructions over a digital elevation model, providing a measure of whether at least three trees are visible to inhabitants.

2. ***30% Canopy – Tree Canopy Coverage Calculation***

A circular buffer, with a predefined radius, is generated around each residential address (or building). The algorithm then calculates the share of the area within the buffer that is covered by tree canopies.

3. ***300 Meters – Proximity to Public Green Spaces***

To assess whether residents have adequate access to public green spaces, a detailed analysis is performed at the level of individual residential addresses or buildings. For each address (or building), the algorithm checks whether it falls within a 300-meter walking distance from a green area.

Since access to green spaces is not always direct or clearly marked, the model incorporates a nuanced definition of access points. Three types of access points are used to capture different possible entryways. The first includes officially designated entrances as mapped in OpenStreetMap (OSM), typically labelled as gates. The second consists of intersections where the boundaries of green spaces meet the pedestrian street network, representing natural points of entry. When neither of these is present, the model generates fictitious access points at regular 100-meter intervals along the perimeter of green spaces, under the assumption that the entire edge is accessible to pedestrians.

4.5.5.3 Technical implementation

The **input data** required are:

- **Tree crowns:** single tree crowns (vector, polygons), tree canopy (vector, polygons), or tree census (vector, points);
- **Residential addresses or buildings**.
- Digital Surface Model **(DSM)** with at least 1 m resolution (raster);
- **Green urban public spaces**.
- Gates/accesses to green urban public spaces (vector, points) from OSM (Open Streetmap).
- Road network from OSM (topological line network).

Output data

- **Rule 3 30 300 residential buildings**: buildings with the addition of the following attributes:
- **Rule 3:** a Boolean value (1 if three trees are visible, 0 otherwise).
- **Rule 30:** a true or false value indicating compliance with the 30% canopy coverage threshold and the corresponding percentage value.
- **Rule 300:** a Boolean field indicating whether a green area is present within 300 meters (true/false) and a distance field (in meters) specifying the proximity to the nearest green area (only populated if the Boolean value is true).

For each building, Boolean values indicate whether each individual rule is satisfied. In addition to these, a final aggregated score combines the three rules into a single index. A score of 0 is assigned if none of the three rules—3 trees, 30% canopy coverage and proximity within 300 meters—are met; a score of 1 if only one is satisfied; and a score of 2 if both are fulfilled. If a third additional criterion is also met, the final score is 3.

Rule 3 – Visibility Analysis calculation

The output is generated through a detailed QGIS workflow involving the following steps:

1. **Extraction of Tree Centroids:** A "Single Canopy Tree" vector layer is processed to calculate the centroids of individual trees.
2. **Virtual Window Generation:** Building polygons were converted into perimeter lines representing building facades. Facades adjacent to other buildings — and therefore unlikely to have windows — were excluded. Along the remaining facades, points were generated at regular intervals to simulate window positions.
3. **Viewshed Analysis:** Using QGIS's Viewshed Analysis tool, visibility from each tree centroid is computed. The analysis determines the surface area visible from a given observation point, based on a digital surface model (DSM) within a radius of 50 meters. The output of the viewshed analysis is a Boolean value: 1 if there are trees visible inside the area, and 0 if not.
4. **Sampling and Output:** The Boolean output from the viewshed analysis was spatially joined with the points generated along the building facades (virtual windows). For each virtual window point, it was determined whether it falls within an area where trees are visible (True) or not (False).
5. **Aggregation on buildings:** The aggregation at the building level is done based on the majority of the generated virtual points being true, producing a Boolean indicator (true/false) to specify whether the rule is satisfied or not.

Rule 30 – Tree Canopy Coverage Calculation

The method relies on vector data representing individual tree crowns or aggregated tree canopy to identify areas of high vegetation. It can be applied using two alternative approaches depending on data availability. In one approach, residential addresses or buildings are used as reference points. For each building or address, a circular buffer zone is generated (e.g., with a radius of 200 or 500 meters). Within this buffer, the algorithm calculates the percentage of the area covered by tree canopies.

Each address or building is then assigned two attributes from this calculation:

- a True/False value indicating whether the canopy coverage meets or exceeds the 30% threshold.
- an integer value representing the percentage of canopy coverage.

This procedure is implemented using a QGIS Modeler workflow. The algorithm is extended to include aggregation at the addresses/buildings level to specify whether the rule is satisfied or not.

Rule 300 - Proximity to Public Green Spaces Calculation

The first phase of the analysis, *Gate Classification*, focuses on identifying and categorizing all potential access points to public green spaces. This step uses three input layers: polygons representing green areas, a set of potential gate locations, and street network data.

A QGIS model classifies each gate into one of three categories:

- **Type A**: official gates: access points explicitly designated in OpenStreetMap (OSM) as formal entrances.
- **Type B: i**ntersections with roads.
- **Type C**: virtual access points.

The second step involves a Python script that processes two GeoPackage files: one containing residential address points (or buildings) and the other containing green space gates. Both layers must use metric coordinate reference systems (CRS) to ensure accurate spatial computations. The script creates a 300-meter circular buffer around each address point. Through a spatial join, the script identifies all gates located within each buffer zone. For every address-gate pair found, the script dynamically constructs a routing request URL with origin and destination coordinates, which is sent to the OSRM API to calculate the true walking distance. If the distance is 300 meters or less, the address is flagged as accessible within the threshold. To optimize processing time, once a qualifying gate is found for a given address, the script skips checking any remaining gates associated with that address. The algorithm is extended to include aggregation at the addresses level to specify whether the rule is satisfied or not.

Although every implementation has primarily utilized OpenStreetMap and Urban Atlas data, the methodology remains compatible with any local or regional datasets.

4.5.6 Public Transport Accessibility Analysis

4.5.6.1 Description and Purpose

The Public Transport Accessibility analysis aims to analyse the impact of the public transportation network on the city's functions. Public transport accessibility plays a vital role in the overall functionality of a city by ensuring that people can move efficiently, affordably, and sustainably between different areas. It enables equitable access to jobs, education, healthcare, and social activities, particularly for those without private vehicles, such as low-income residents, the elderly, and youth. High accessibility reduces traffic congestion, lowers greenhouse gas emissions, and enhances urban liveability by encouraging a shift away from car dependency. Furthermore, it supports economic

development by connecting businesses to a wider labour pool and customer base. In essence, accessible public transport is a cornerstone of inclusive, resilient, and well-functioning urban systems.

4.5.6.2 Key Features and Functionality

This analysis aims to comprehensively quantify the quality and effectiveness of a city's public transportation network, focusing on two key dimensions. The first dimension evaluates the spatial distribution and density of public transport access points—such as bus stops, train stations, and tram lines—across the urban landscape. Proximity to these access points is a critical factor influencing citizens' likelihood of choosing public transport for their daily commutes, especially when convenience and time savings are at stake. The second dimension assesses the network's reachability, which refers to how efficiently different parts of the city are connected through available public transport modes. This includes analysing travel times, transfer frequency, and network coverage to determine whether residents can access essential services, workplaces, and social opportunities without excessive detours or delays. Together, these axes provide a holistic view of accessibility, highlighting both the physical presence and the functional performance of public transport within the city.

4.5.6.3 Technical implementation

This analysis will operate in a manner like the 15-minute city index described in Section 4.5.7. The city's road-network topology will be assessed to extract road intersections, which serve as access points connecting each city block to the rest of the urban fabric. This topology is downloaded from open-source repositories such as OpenStreetMap, and the precise locations of public-transportation points are retrieved from the same source. The analysis encompasses various modes of public transport—including bus stops, train stations, tram lines, and metro lines. Next, the tool calculates the proximity of each city block to these modes, assigning a weighting factor to each to reflect its contribution to the overall system. For example, in smaller cities buses may be more important than trains for everyday trips, whereas in larger cities metro lines usually form the backbone of the network. The second part of the analysis incorporates public-transport schedules and routes to determine how easily different parts of the city can be reached from a given starting point. The final output is a Public Transport Accessibility Index which, like the 15-minute city analysis, is presented as a visual map overlay showing the scores for the various parts of the city.

In terms of technology stack, this module is python-based, utilizing similar libraries to those described in Section 4.5.7.3. Given that the development of the module is still ongoing, its integration in the URBREATH Toolbox has not commenced, and will be described in detail in the next deliverable version.

4.5.7 15 minutes city (proximity index)

4.5.7.1 Description and Purpose

The algorithm is designed to assign a proximity index to services within a specific territory, evaluating how easily points of interest (PoIs) can be reached on foot or by bicycle, in line with the "15-minute city" concept introduced by Carlos Moreno. This urban model envisions that most daily needs should be met within a 15-minute walk or bike ride from home. The core aim is to measure the accessibility of

essential urban services in a way that is both globally applicable and locally detailed. To ensure broad usability and replicability, the algorithm relies exclusively on freely available, open-source data.

4.5.7.2 Key Features and Functionality

Algorithm Steps

1. Tiling the Area of Interest

The area is divided into rectangular tiles based on a maximum size specified by the user. The algorithm automatically calculates a uniform tile size that fits the entire area, ensuring all tiles are equal in size and as close as possible to—but not exceeding—the user-defined maximum. This step is performed only once, while the subsequent steps are repeated for each tile.

2. Data Download

For each tile, the algorithm starts the process by creating the necessary folders to store the downloaded data. It then retrieves all essential datasets required for the analysis. If elevation analysis is enabled, a Digital Elevation Model (DEM) is first downloaded to incorporate topographic context. This is followed by downloading the street network from OpenStreetMap, which provides the basis for calculating routes and accessibility. Lastly, the algorithm collects the Points of Interest (PoIs) corresponding to the selected service categories.

To ensure that the spatial coverage is complete—especially along the edges of each tile—the area from which data is downloaded is intentionally expanded beyond the tile boundaries. Specifically, the download area extends an additional 4 kilometres for walking analyses and 9 kilometres for cycling, guaranteeing that the proximity calculations remain accurate even near the edges of each tile.

3. Generating the Hexagonal Grid (Computation step)

Each tile is now divided into hexagons with a diameter of 250m (125 m side), chosen as the optimal trade-off between spatial resolution and computation time.

4. Calculating the Proximity Index

The index is calculated based on the walking time from nodes in the street network to the nearest point of interest (PoI) within each service category, considering only streets that are accessible to pedestrians. Each hexagonal unit is then assigned a proximity value that reflects two key aspects: the presence of essential services in the surrounding area—covering eight categories including groceries, education, healthcare, banks and post offices, parks, entertainment, sustenance, and shops—and the ease of reaching those services via paths that are either walkable or bikeable, depending on the selected mode of travel (either on foot or by bicycle).

Each hexagon receives one of the following values:

1. **15-minute zone** – all main services reachable within 15 minutes.
2. **30-minute zone** – at least one service takes between 15 and 30 minutes to reach.

3. **60-minute zone** – at least one service takes between 30 and 60 minutes.
4. **Low-proximity zone** – at least one service is not reachable within 60 minutes.
5. **No-proximity zone** – none of the services are reachable within 60 minutes.

5. Merging Results

Once the analysis is completed for each individual tile, the results are merged into a single output file. This consolidated file, named according to the selected mode of travel (e.g., walkability_foot.csv or walkability_bike.csv), is saved in the specified output folder. The spatial data within the file is referenced using the EPSG:3857 coordinate system.

Orography Consideration

The algorithm can account for terrain elevation, resulting in more realistic travel time estimations by adjusting pedestrian speed based on slope. This improves accuracy compared to the traditional approach that assumes flat terrain and constant walking speed.

The algorithm is executed using the following command:

versionePy esecutore.py "[latMin, lonMin, latMax, lonMax]" diamEsag dimMaxTass "pathOutFolder" Orography weight category by > pathOutFile 2>&1 &

Command parameters:

- **"[latMin, lonMin, latMax, lonMax]":** bounding box of the area of interest, in EPSG:4326**.**
- **diamEsag:** diameter of the hexagon, the base spatial unit used for aggregation (250m)**;**
- **dimMaxTass:** maximum size of the tiles into which the area is divided**.**
- **"pathOutFolder":** folder path where the output will be saved.
- **Orography (default: 'False'):** True or False, specifies whether to account for terrain elevation.
- **weight:** impedance type to define distances on the network (e.g., 'space', 'time').
- **category (default: 'all'):** the service category on which to calculate the index (one of the 8 available service categories).
- **by (default = 'foot'):** travel mode used to compute the index ('foot' or 'bike').

4.5.7.3 Technical implementation

The algorithm has been implemented in Python and relies on a set of specialized libraries for geospatial analysis, network modelling, and data handling. It can be executed in any standard Python environment equipped with an IDE or via command line and requires access to OpenStreetMap for data retrieval.

Table 9: 15 minutes city – technical background

Programming language	Python
Libraries used	Pandana, geopandas, numpy, pandas, osmnet, rtree, pyproj, shapely, geovoronoi , fiona=1.9.5, rasterio, gdal,scipy, beautifulsoup4
Tools needed to implement and run the algorithm	Python, IDE, command line

Input and Output Data

Input data is downloaded from OpenStreetMap using the pandana.loaders library. The data is saved as CSV files in the output folder.

Input data is divided into two categories:

- **Network** (nodes and edges and their length);
- **Pois** (points of interest), divided into 8 categories based on the OSM Wiki classification.

Road network data is requested with the filter for "network type" set to "walk", with the effect of downloading only walkable streets.

This step can be replaced with a custom alternative process in case the user plans to use data from a source different from OSM.

Output data is in CSV format with the following columns:

- **Geometry:** hexagon WKT.
- One column per each poi category showing the **average** time to the nearest poi.
- **countNaN:** number of categories not reachable within 60 minutes on foot.
- **Average** time to reachable Pois (only those within 60 minutes).
- **City:** general proximity level of the hexagon (one of the 5 levels).

4.5.8 Visual Interpretable and Explainable AI

4.5.8.1 Description and Purpose

In recent years, Explainable Artificial Intelligence (XAI) has emerged as a critical field within the broader AI domain, aiming to demystify the inner workings of complex machine learning models. As AI systems are used in critical areas like healthcare and urban planning, XAI helps build trust by clarifying how inputs lead to specific outputs. This allows both experts and non-experts to interpret, validate, and respond to AI-driven decisions with greater confidence and accountability. In the context of the URBREATH project, the Visual Interpretable and Explainable AI (VIE-AI) tool developed by EXUS, plays a key role in facilitating a better understanding of the decision-making processes embedded in advanced AI systems. Its primary function is to take as input the AI models developed by other project partners, along with a representative subset of the datasets used to train those models.

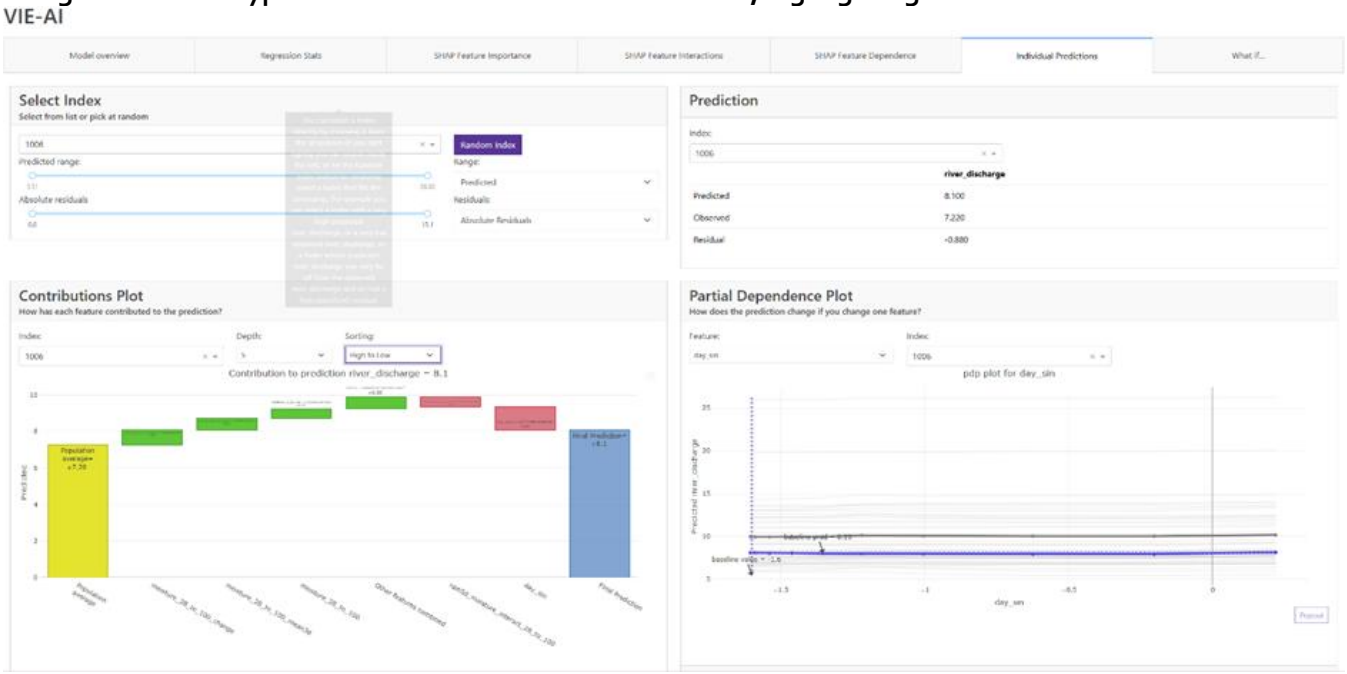
This approach allows VIE-AI to generate meaningful insights into how various input features affect model predictions.

At its core, VIE-AI leverages a suite of explainability techniques tailored to different user needs. One of its central features is Feature Importance, which enables the visualization of how much different input variables contribute to a model's predictions. Additionally, the tool implements SHAP (SHapley Additive exPlanations) values, a state-of-the-art method for attributing model predictions to individual input features in a mathematically sound and visually intuitive way. This allows users to assess not only which features are important, but also the direction and magnitude of their influence on specific predictions. While the full potential of VIE-AI will be realized when it is applied to all AI models developed under the URBREATH project, its current application has focused on a pre-existing model provided by EXUS. This initial demonstration has successfully showcased VIE-AI's core capabilities. As other models from project partners become available, VIE-AI will be systematically integrated to support transparency and interpretability across the project's AI-based analytics layer.

4.5.8.2 Key Features and Functionality

The VIE-AI tool integrates a wide array of features designed to enhance model transparency and support deep interpretability of AI systems. One of its central components is the "Basic Model Understanding" section, which provides users—particularly data scientists and technical experts—with essential insights into the nature of the model being analysed. This includes a summary of the model's type (e.g., regression or classification), key performance metrics, and visual diagnostics such as observed vs. predicted value plots and residual plots. These visualizations help users quickly assess the reliability and performance of the model by highlighting where and how predictions diverge from observed outcomes.

Figure 12: Prototyped version of the VIE-AI tool interface, highlighting the *Individual Predictions* tab.



A core pillar of VIE-AI is its integration of SHAP analysis. The tool offers multiple SHAP-based tabs, including feature importance plots where input variables are ranked according to their contribution to model predictions. Users can sort these visualizations based on absolute SHAP values or permutation importance and customize the number of features displayed. Additional SHAP functionalities include interaction plots, which allow detailed inspection of relationships between pairs of features, and feature dependency plots, which reveal nonlinearities and thresholds that may influence outcomes. Complementing the SHAP analysis is the inclusion of LIME (Local Interpretable Model-agnostic Explanations). This section of the tool mirrors SHAP's objectives but uses alternative visualization techniques, such as heatmaps and case-based analysis of prediction contributions. These allow users to compare actual versus predicted outcomes on an individual basis and examine how specific features influence a single prediction.

Interactivity is a strong theme throughout the VIE-AI dashboard. The "Individual Predictions" tab (see Figure 12) enables users to select specific instances or time points to examine in detail. This includes viewing prediction errors and the cumulative contribution of features toward a specific output. Another noteworthy feature is the Partial Dependence Plot (PDP) functionality. Users can adjust sampling rates and visual parameters to analyse how changes in one input feature affect model predictions on average. The tool also includes a contribution table that lists features from most to least impactful for a selected instance, further supporting localized explanation.

4.5.8.3 Technical implementation

As previously mentioned, the VIE-AI tool implements a model-agnostic explainability framework designed to work with pre-trained machine learning models in pickle format. The system takes as input trained ML models and representative testing datasets, then provides comprehensive explainable AI capabilities without requiring access to the original training pipeline or data sources.

System Components

- **Model loading & validation layer -** Handles ingestion of pre-trained ML models from pickle files, validates model compatibility, and extracts model metadata and feature requirements.
- **Data processing layer** - Processes testing datasets to match model input requirements, handles feature alignment, and prepares data for explainability analysis.
- **Explainability engine** - Core XAI functionality through model-agnostic RegressionExplainer / ClassificationExplainer, SHAP/LIME integration for feature attribution, and explanation generation without requiring original training data
- **Visualization layer** - Custom Plotly components for model performance analysis, prediction explanations, and interactive dashboards with specialized tabs (feature importance, interactions, dependence plots, what-if analysis).

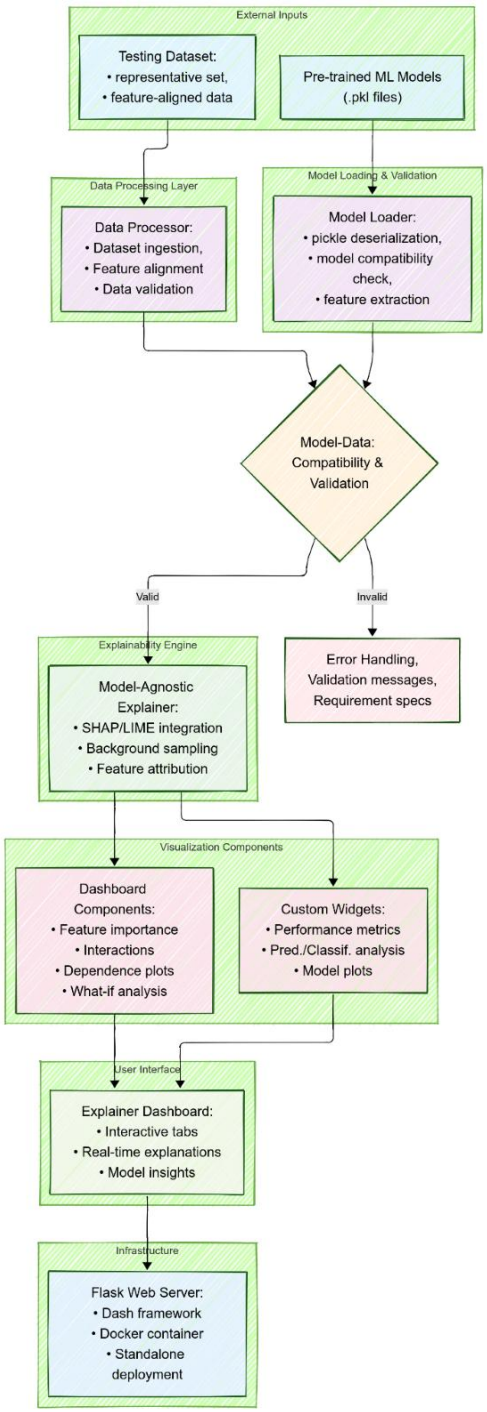
Core technology stack

Backend components:

- **Python 3.10.15**: Core runtime environment

- **Scikit-learn**: for Machine Learning pipeline implementation
- **SHAP/ LIME**: explainability models for feature attribution
- **ExplainerDashboard**: framework for building interactive ML model dashboards
- **Pandas & NumPy**: data manipulation and numerical computing

Figure 13: VIE-AI System Architecture Diagram



Frontend & visualization:

- **Dash/Plotly**: interactive web application framework and visualization library
- **Dash Bootstrap Components**: UI component library for responsive layouts
- **HTML/CSS**: custom styling and layout management

Component communication

The system follows a streamlined workflow as depicted in **Figure** 13: pre-trained model loading → test dataset ingestion → model-data compatibility validation → explainer initialization with background sampling → interactive dashboard generation → real-time explanation computation for user interactions.

4.5.9 Tree growing prediction

4.5.9.1 Description and Purpose

This plugin of the VC Map simulates the growth of trees over a set number of years, allowing users to adjust growth rates and visualize annual changes. It provides detailed reports on tree growth, including canopy coverage, oxygen production, CO_2 sequestration, and water demand. The simulation considers different tree species, growth rates (slow, moderate, fast), and age transitions (e.g., young to middle-aged). Results are presented in tables and charts, summarizing key metrics such as total and average values for oxygen, CO_2, and water. The plugin also generates PDF reports with simulation data and visualizations. Growth is modelled linearly per year, with special logic for lollipop trees and no negative growth. Environmental factors like weather or soil are not included. The calculations are based on typical values for urban and temperate trees. The plugin estimates the number of people supported by the produced oxygen. All simulation results are based on the current configuration and assumptions outlined in the summary.

4.5.9.2 Key Features and Functionality

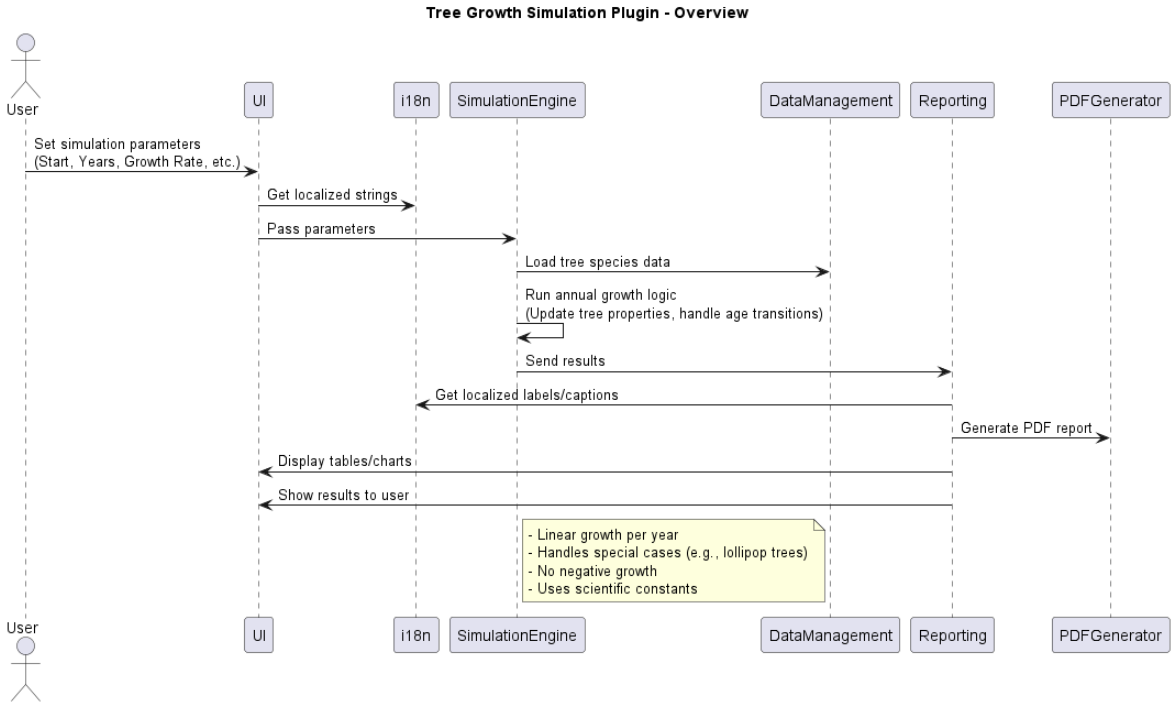
- Simulates tree growth over a user-defined number of years, with adjustable growth rates (slow, moderate, fast).
- Allows dynamic visualization of annual growth, including the option for time-delayed simulation steps.
- Provides detailed reports and summaries, including tables and charts for canopy coverage, oxygen production, CO_2 sequestration, water demand, and biomass.
- Supports PDF report generation with simulation results, diagrams, and calculation area screenshots.
- Calculates and displays both total and average values for key metrics (oxygen, CO_2, water) per year and per tree species.
- Models tree age transitions (e.g., young to middle-aged) and handles special cases like lollipop trees.
- Excludes negative growth and environmental factors (e.g., weather, soil) for simplicity.
- Uses scientifically based assumptions for growth rates, oxygen production, CO_2 sequestration, and water needs.

- Estimates the number of people supported by the produced oxygen.
- All results and calculations are based on the current simulation settings and are clearly documented in the summary.

4.5.9.3 Technical implementation

- The plugin uses a structured JSON-based internationalization (i18n) system, to manage all user-facing text and labels, supporting easy localization.
- The simulation logic is likely implemented in TypeScript
- The UI is built with Vue.js, enabling reactive and interactive user interfaces.
- Simulation parameters (growth rate, years, dynamic growth, etc.) are configurable via the UI and passed to the simulation engine.
- The plugin generates both tabular and graphical outputs, with chart titles and axis labels defined in the i18n files, suggesting integration with a charting library.
- PDF report generation is supported, with sections and captions defined in the i18n structure, likely using a PDF generation library in the backend.
- The simulation models tree growth annually, updating tree properties each year and handling transitions between age classes and special cases (like lollipop trees).
- All calculations are based on scientific assumptions and constants, which are documented in the summary and likely implemented in helper or constants files.
- The codebase is modular, separating concerns such as simulation logic, reporting, UI, and localization for maintainability and scalability.
- The plugin does not account for environmental variability, focusing on deterministic, data-driven growth modelling for clarity and reproducibility.

Figure 14: Tree growing prediction - sequence diagram



Below you can find some images, taken from the report of the plugin:

Figure 15: Tree growing prediction – example of report 1 of 3

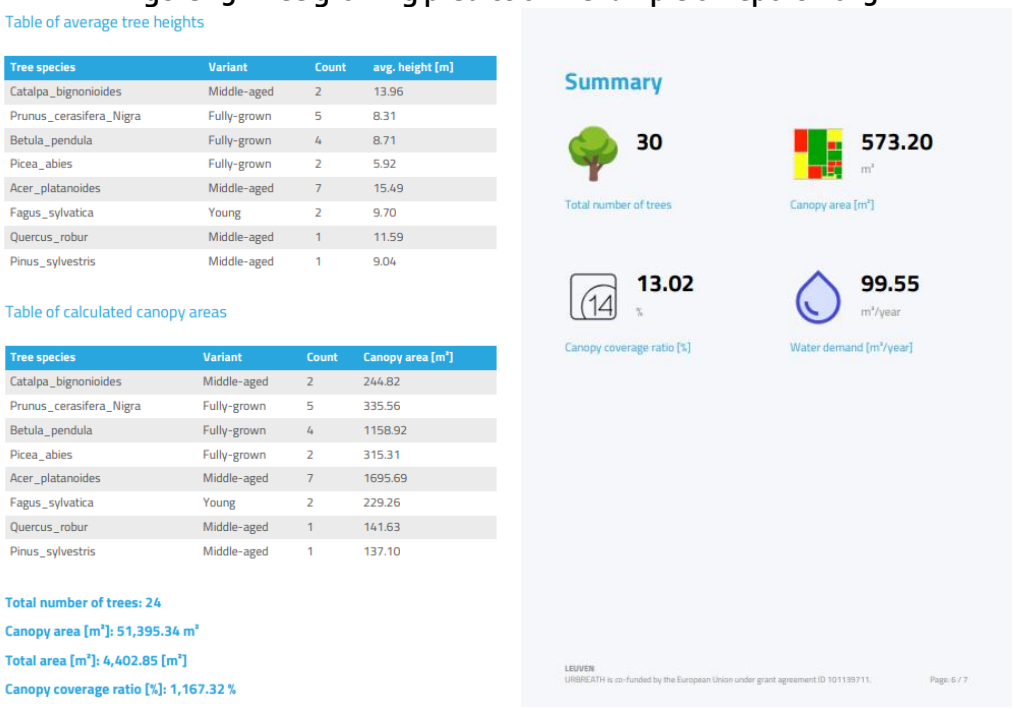


Table of average tree heights

Tree species	Variant	Count	avg. height [m]
Catalpa_bignonioides	Middle-aged	2	13.96
Prunus_cerasifera_Nigra	Fully-grown	5	8.31
Betula_pendula	Fully-grown	4	8.71
Picea_abies	Fully-grown	2	5.92
Acer_platanoides	Middle-aged	7	15.49
Fagus_sylvatica	Young	2	9.70
Quercus_robur	Middle-aged	1	11.59
Pinus_sylvestris	Middle-aged	1	9.04

Table of calculated canopy areas

Tree species	Variant	Count	Canopy area [m²]
Catalpa_bignonioides	Middle-aged	2	244.82
Prunus_cerasifera_Nigra	Fully-grown	5	335.56
Betula_pendula	Fully-grown	4	1158.92
Picea_abies	Fully-grown	2	315.31
Acer_platanoides	Middle-aged	7	1695.69
Fagus_sylvatica	Young	2	229.26
Quercus_robur	Middle-aged	1	141.63
Pinus_sylvestris	Middle-aged	1	137.10

Total number of trees: 24

Canopy area [m²]: 51,395.34 m²

Total area [m²]: 4,402.85 [m²]

Canopy coverage ratio [%]: 1,167.32 %

Figure 16: Tree growing prediction – example of report 2 of 3

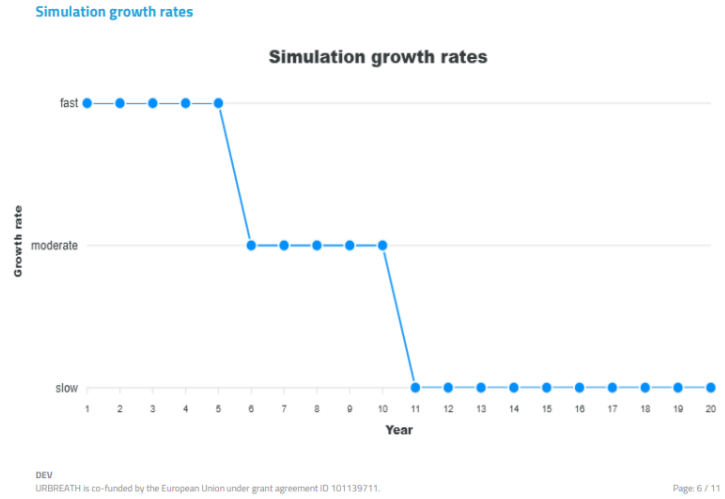
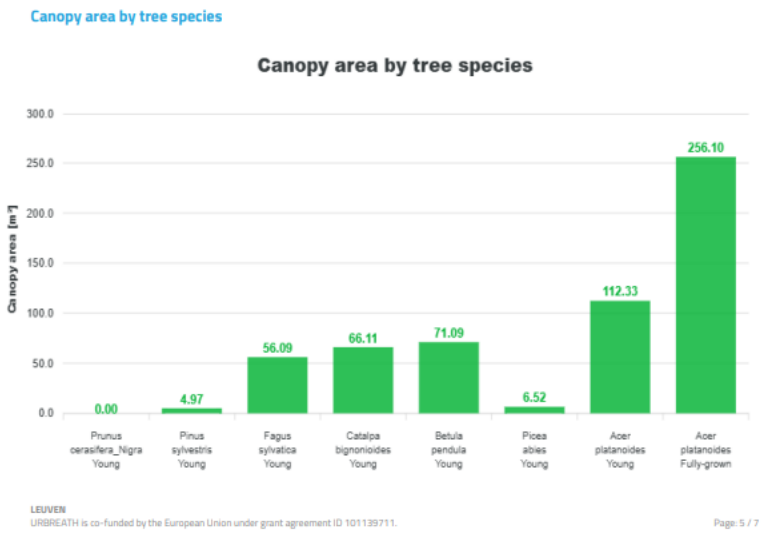


Figure 17: Tree growing prediction – example of report 3 of 3



4.6 City Monitoring

4.6.1 KPI Manager

4.6.1.1 Description and Purpose

The **KPI Manager** tool is designed to monitor, analyse, and manage key performance indicators at various organizational levels. Its main objective is to provide useful information that supports strategic decision-making and performance optimization.

The tool aggregates information from various sources, including REST APIs, disparate databases and frost server, allows users to define KPIs tailored to their specific objectives and tracks them over time.

Through intuitive visualization tools, such as dashboards, tables, and graphs, the tool offers users a clear overview of KPIs' progress.

4.6.1.2 Key Features and Functionality

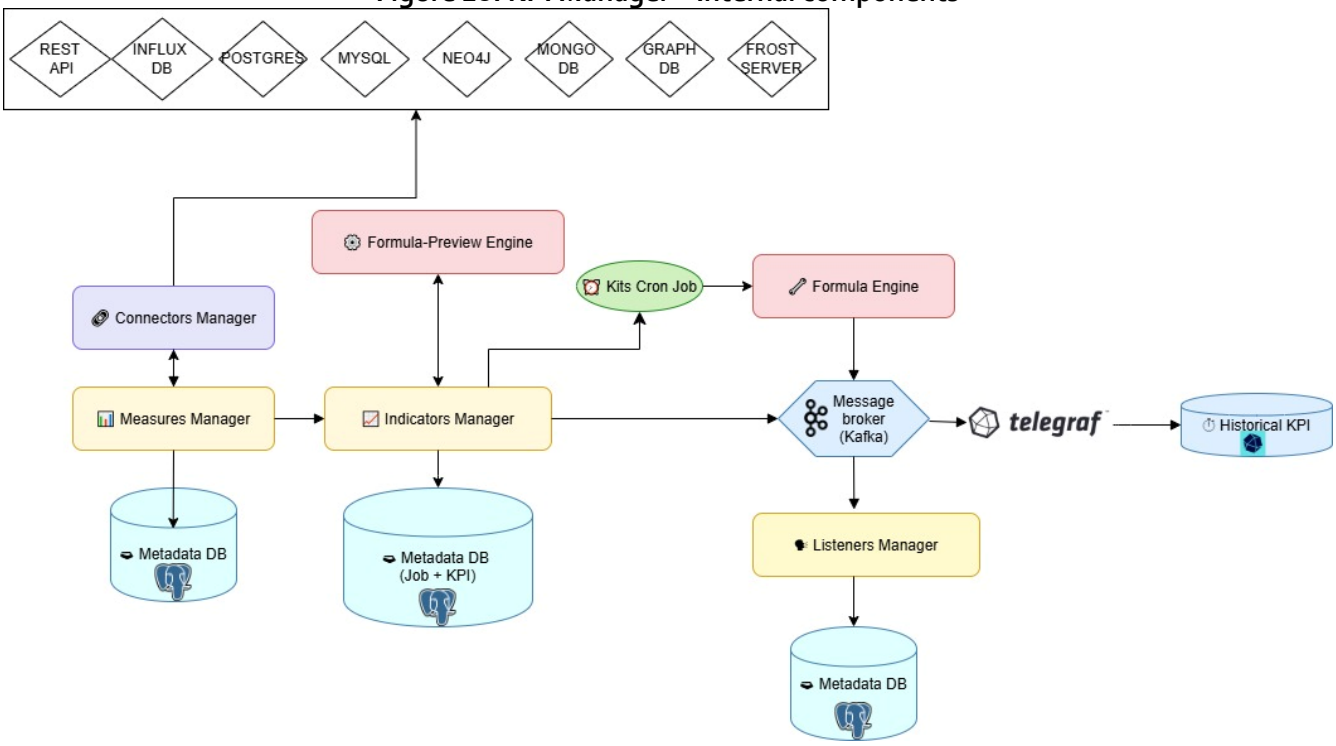
This tool offers many functionalities categorizable as follows:

- Measurement Definition: ability to aggregate data from various sources, such as REST APIs, different databases and IT systems, such as Frost Server.
- Customizable KPIs: define and manage performance indicators based upon specific business objectives.
- Target Setting: set performance targets and identify deviations.
- Visual Representations: charts and graphs allow users to identify trends, patterns, and anomalies.
- Historical Monitoring: track KPIs' evolution over time to evaluate trends and the impact of interventions.
- Collaboration Tools: share charts, graphs, reports, and KPI data with stakeholders to support transparency and collective goal tracking.
- Insight Generation: provides real-time insights to guide timely decision-making and proactive management.

4.6.1.3 Technical implementation

The KPI Manager is designed using a microservices-based pattern for both frontend (Vue.js, Single-SPA) and backend (NodeJS, TypeScript, InfluxDB, PostgreSQL) components, ensuring modularity, scalability, and ease of maintenance. Communication between the two sides of the application is handled by a middleware microservice managing authentication and validation.

Figure 18: KPI Manager – Internal components



The KPI Manager is organized into a series of specialized microservices, coordinated by a central **middleware** (not depicted in the figure above to reduce the complexity of the diagram) that acts as a single point of access for all users. This middleware is responsible for authentication and access authorization, managed through Keycloak. Keycloak is also used to organize user contexts, implemented as groups, thus facilitating granular control of permissions and multi-tenancy access. In addition, the middleware validates incoming requests and provides administrators with dedicated tools for configuring the entire ecosystem.

At the heart of data collection is the **Measure Manager**, a microservice that enables complete management (creation, reading, updating, and deletion) of "Measure" entities. These entities contain all the information necessary to connect to heterogeneous data sources, such as REST APIs, MongoDB databases, SQL, InfluxDB, and are governed by role and context-based security mechanisms.

Once the measures have been defined, the **Connectors Manager** comes into play, retrieving the actual data from these sources using the measure IDs provided by the user. The collected data is then structured and transmitted through the Message Broker (i.e. Kafka; 4.3.1), the messaging system that connects the various microservices.

The actual calculation of KPIs is handled by the **Indicator Manager**, which allows to configure indicators, defining formulas and setting jobs for their periodic execution. These jobs are instantiated as CronJobs on Kubernetes, using the **Formula Engine**.

The Formula Engine actually executes the KPI formulas. This microservice integrates with Measure Manager and Connectors Manager to collect current data, but also with Listeners Manager to obtain historical KPIs, if required by the formula. Once the data has been collected, the formula is evaluated, and the result is sent again via Kafka for storage.

Once executed, the results of the jobs are sent via Kafka and managed by Telegraf[25], which stores them in the Timeseries database (i.e. InfluxDB; section 4.1.2); Telegraf allows easy integration with multiple sources and destinations, simplifying data monitoring and reducing the need to develop additional components.

Towards interconnection with other tool, the **Listeners Manager** provides an interface that allows external systems to access historical KPI data stored in InfluxDB. This is essential for retrospective analysis, reporting, and performance benchmarking.

The KPI manager leverage Kubernetes[26] to ensure scalability and the isolation of user data. Within the Kubernetes cluster, a service account must be configured with the necessary permissions to create Jobs and CronJobs[27], enabling KPI calculations to be performed according to the defined schedule.

KPI Manager is accessible through a Web user interface, which offers two main sections: the measures section and the KPIs section. Measures are needed as units thorough which obtain data used in the monitoring of the KPIs.

After logging into the platform, the user accesses the "Measure" section (Figure 19: KPI Manager - Creation of a measure), which is dedicated to configuring measures. Here the user has the option of creating a new measure by choosing a descriptive name that clearly identifies it in the system.

In the example depicted in Figure 19 the selected measure type is an external source via REST API. The user sets the HTTP method (in this case GET), then enters the URL of the API from which he wants to retrieve the data. At this point, he specifies a JSONPath, which is the exact path in the body of the JSON response where the value to be measured is located.

[25] An open-source data collection agent; https://docs.influxdata.com/telegraf/v1/
[26] KPI manager can make use also of smaller solution for "standalone" deployment, such as Minikube, a tool that allows to excute Kubernetes on local machines; https://minikube.sigs.k8s.io/docs/
[27] CronJob allows scheduling the execution of tasks (i.e. jobs) on an IT system. Schedules are defined in Cron format; https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

To make the context or goal of the measurement more understandable, the user can add a short description. Immediately after that, the platform offers a preview: a call to the API is made and it shows how much data (or values) were correctly retrieved based on the configuration entered (Figure 20).

If everything is correct, the user saves the new measure, which will then be available for monitoring, benchmarking, or display in the dashboard.

Figure 19: KPI Manager - Creation of a measure

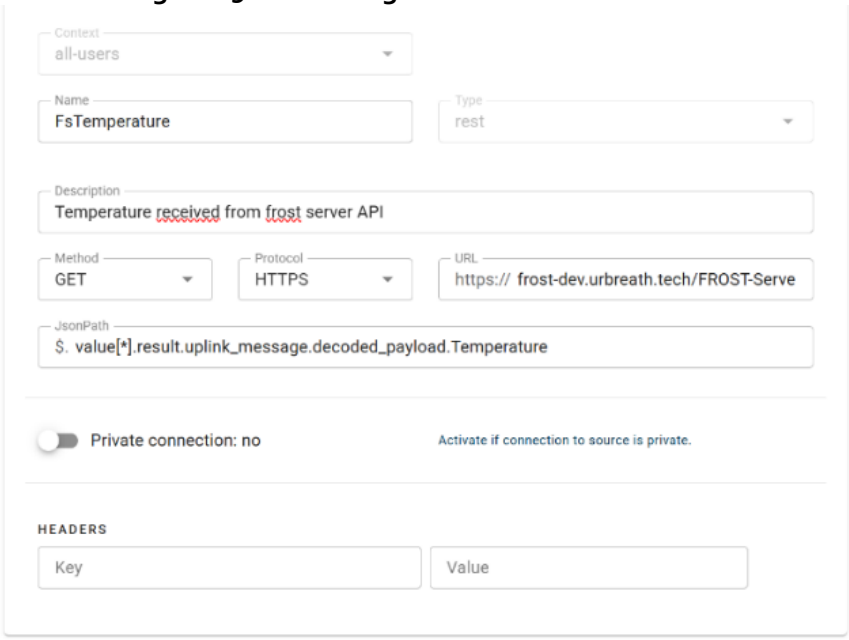
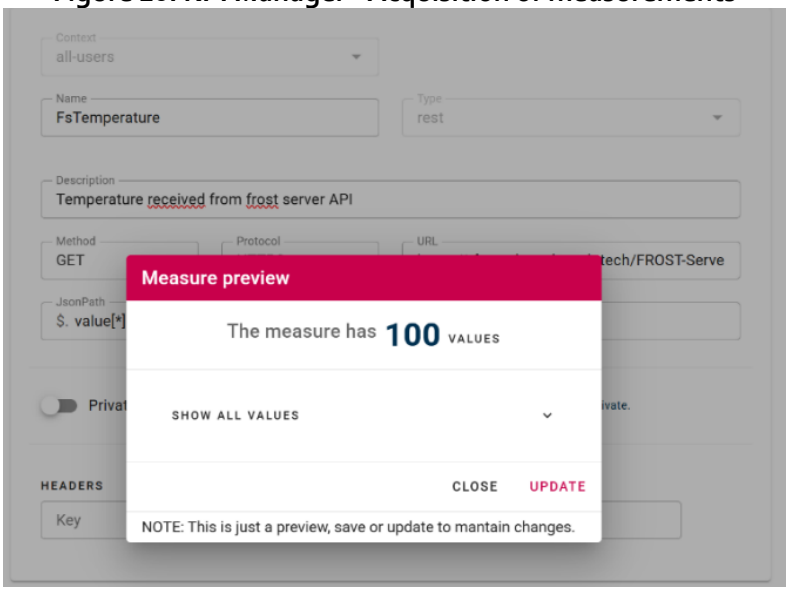


Figure 20: KPI Manager - Acquisition of measurements



After setting up the measures, the user can head to the 'KPI' section to define a new key performance indicator. The first step involves entering the name and description of the KPI (Figure 21). Following that, the user has the possibility to link the KPI to a certain geographic location.

Then the user selects the measures that the KPI will utilise (Figure 22). In the formula section, the user can define the mathematical formula that combines the available measures to define the KPI calculation. Also, the user can preset the schedule for updating the KPI as well as define the target which is the desired value to be reached or sustained (Figure 23).
In the third step, the user chooses the appropriate chart that will best illustrate the KPI visually (line, bar, etc.) (Figure 24). Lastly, the user can preview the result and verify its correctness before saving the KPI (Figure 25).

Figure 21: KPI Manager - Title and description of a KPI

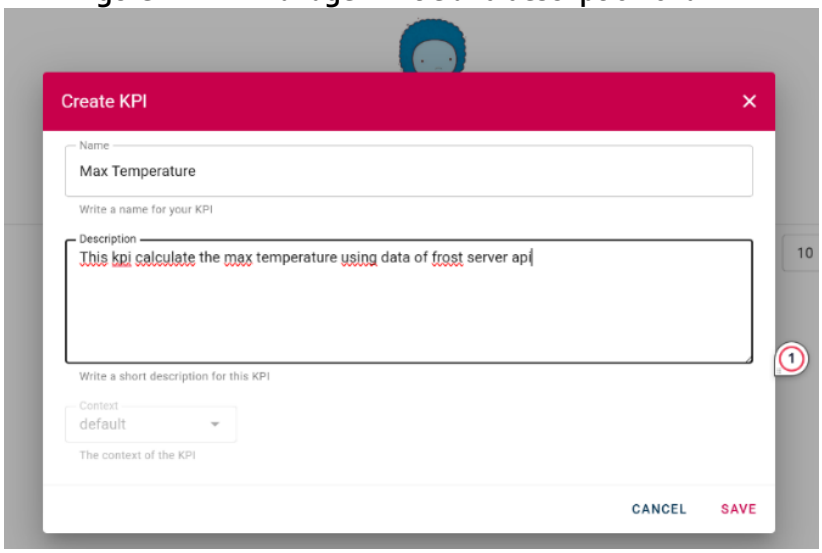


Figure 22: KPI Manager - Selecting the measure to use

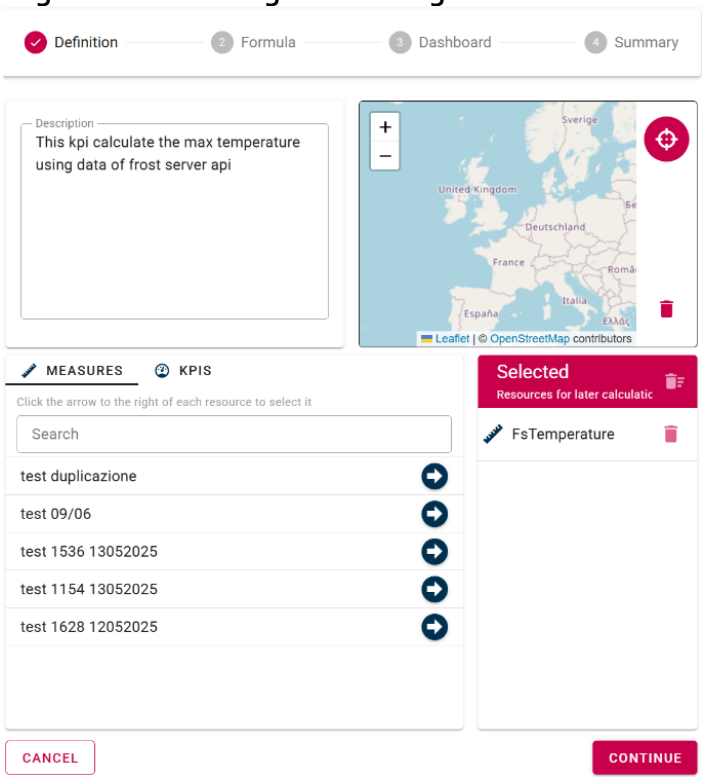


Figure 23:KPI Manager - Insert formula, schedule and target

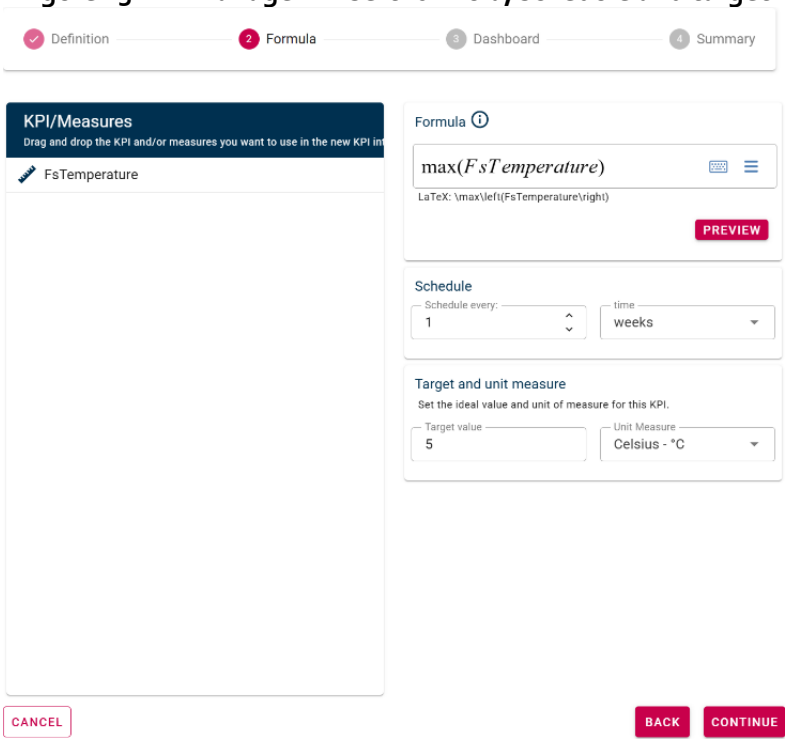


Figure 24:KPI Manager - Select Chart Type

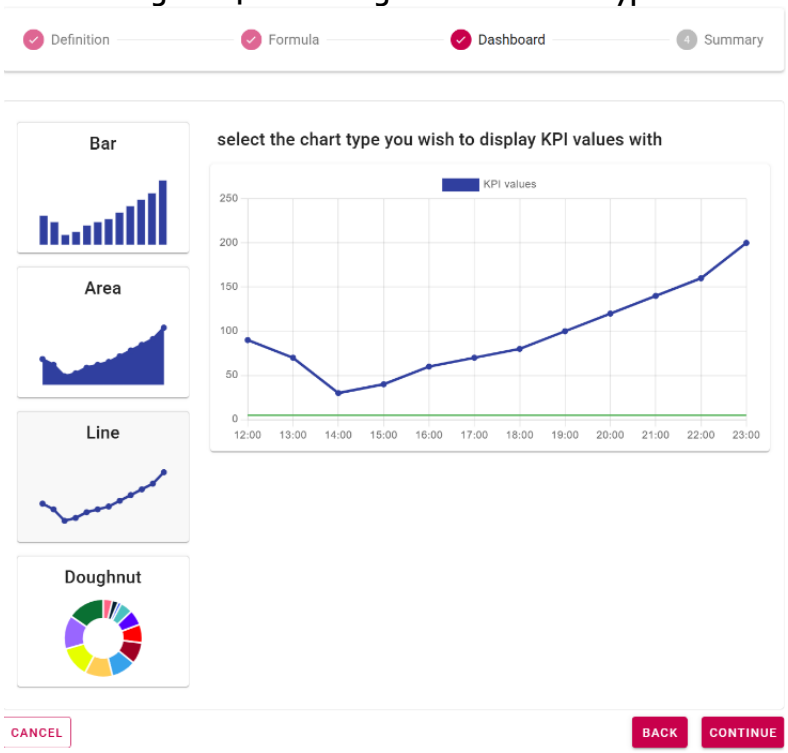
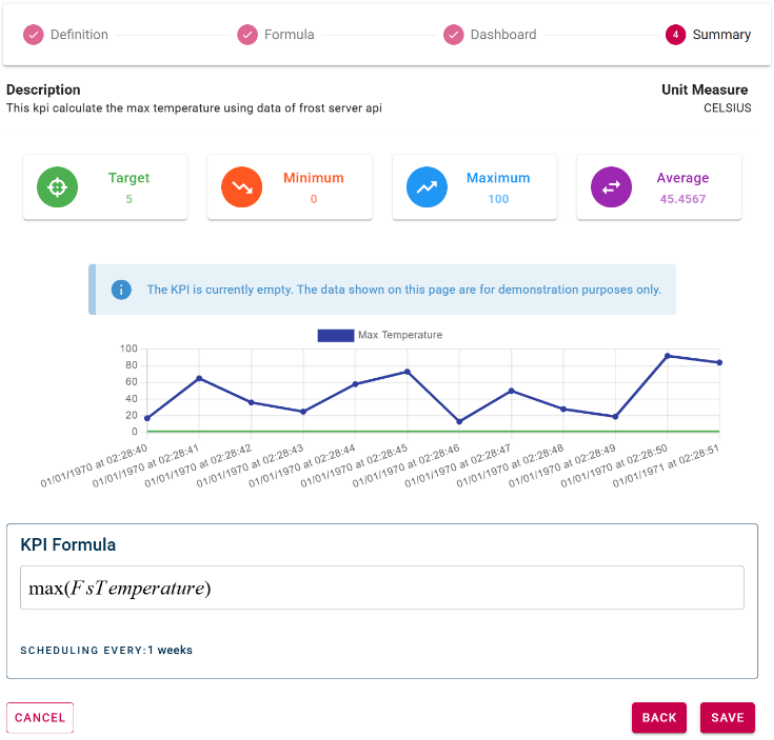


Figure 25: KPI Manager – Save the KPI



4.7 NBS Planning

4.7.1 NBS Registry

4.7.1.1 Description and Purpose

The **Nature-Based Solutions (NBS) Registry** is an application designed to **track, verify, and manage** projects that use nature-based methods to address environmental, social, and climate-related challenges. These solutions can include activities like reforestation, wetland restoration, sustainable agriculture, and conservation that deliver measurable benefits such as carbon sequestration, biodiversity protection, and improved water and air quality.

The **Nature-Based Solutions Registry** serves as a centralized repository for documenting and overseeing NBS projects. Its primary purposes are to:

- **Promote transparency** by publicly listing project data.
- **Track and/or predict environmental outcomes,** such as carbon credit or biodiversity metrics related to the projects.
- **Ensure accountability** through third-party validation and verification of project impacts.
- **Facilitate investment** by providing credible data that stakeholders can trust.

4.7.1.2 Key Features and Functionality

In the current version, the NBS Registry facilitates the following key functionalities:

- **Project Registration**: It allows pilot administrators to register project data including geographical information, media and related material.
- **Public Access & Transparency**: It provides open data access to stakeholders and enables search and visualization of project locations and overall data based on criteria like climate zone, city, keywords and related problems.
- **Integration with tools like the Dataset Catalogue**: It allows a project administrator to corelate the NBSs with relevant datasets.

All users of the system can access the search page of the NBS Registry and look for solutions that match certain criteria (

Figure 26). These criteria relate to the NBSs climate zone, the project city and any problems or keywords the NBS may have. The NBSs are presented on a map to allow the user to view easier the geographical distribution of the NBSs, as well as in a box list form with basic information like the title, a description of the NBS, related keywords, etc.

Clicking on one of the NBSs through the map or the list the user can navigate to the details page and view all the information related to the NBS (Figure 27). A user with admin privileges can also create a new NBS (Figure 28). The creation of the NBS involves the data entry of the NBS in four stages. In the first tab, the user enters the NBS primary data like the title, the climate zone, pilot, status, location, main image and whether it is an URBREATH NBS. After the completion of all the primary data, the user can move on to the next step which involves the entry of the secondary data like the area characterization, the challenges, the objective, the potential benefits and the lessons learned. In the next step the user connects the new NBS with datasets and KPIs from the Data and KPI Catalogues respectively and in the final one he/she can upload photos, videos and related material.

Figure 26: NBS Registry – Search page

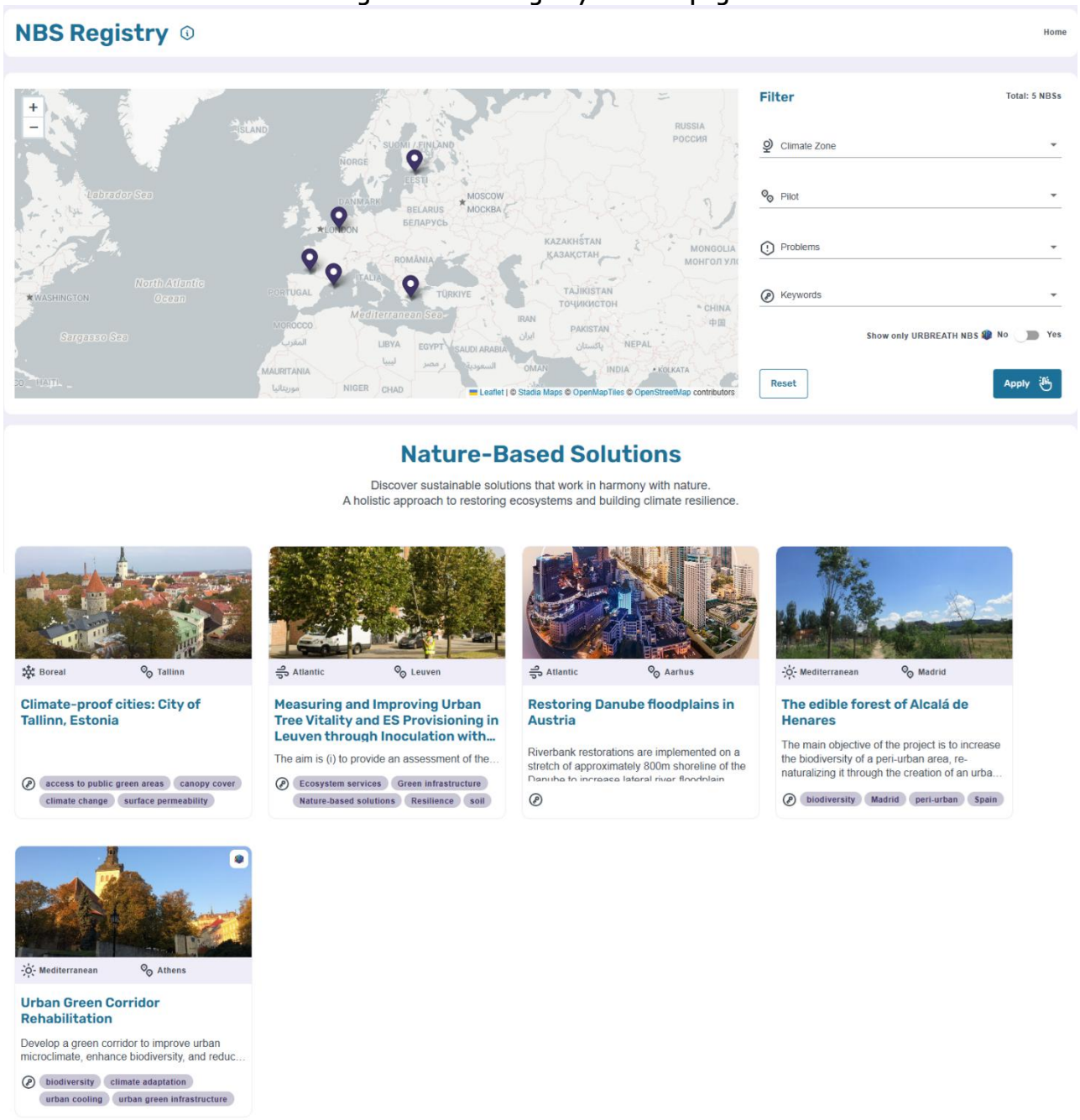


Figure 27: NBS Registry – Details page

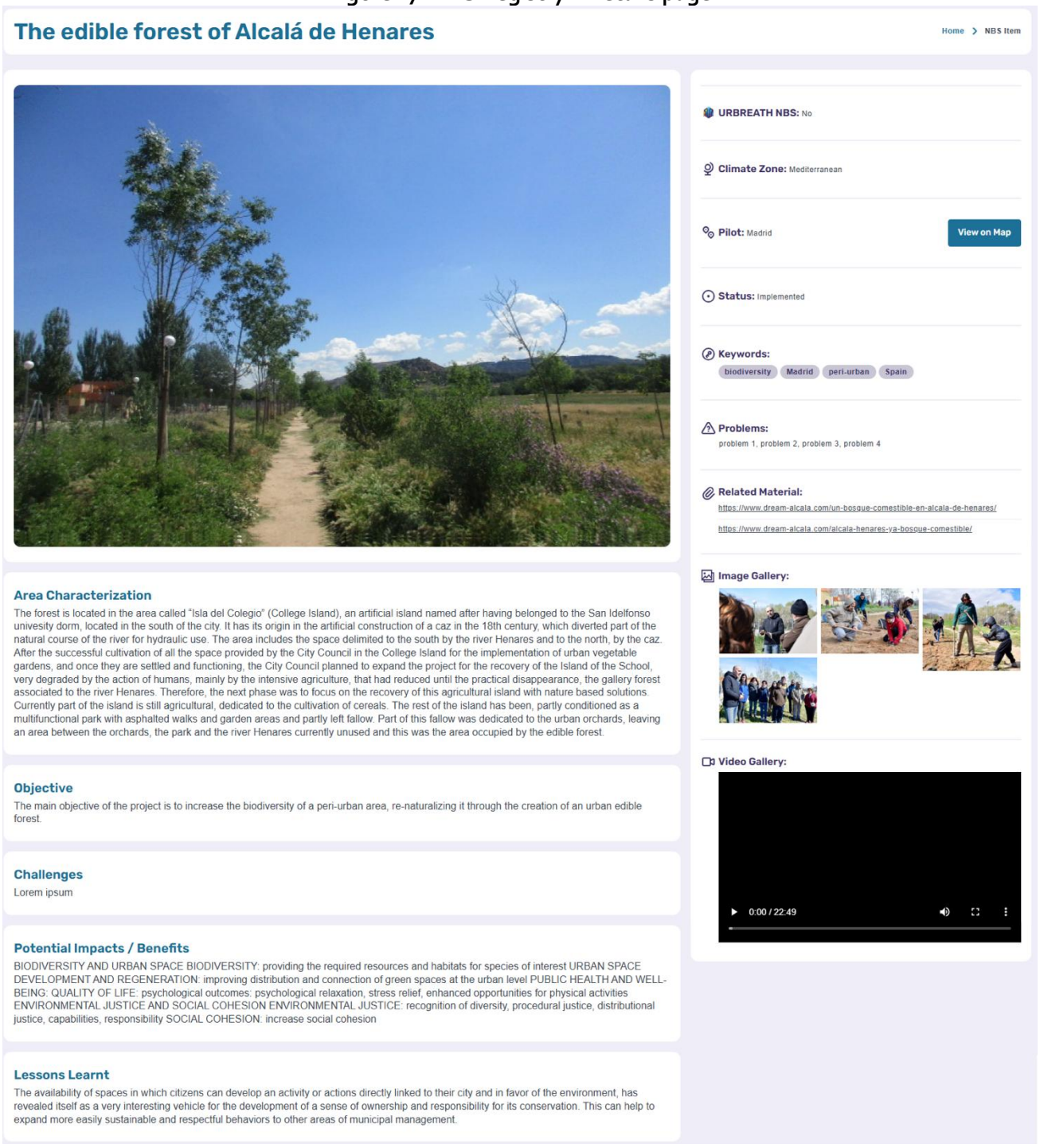
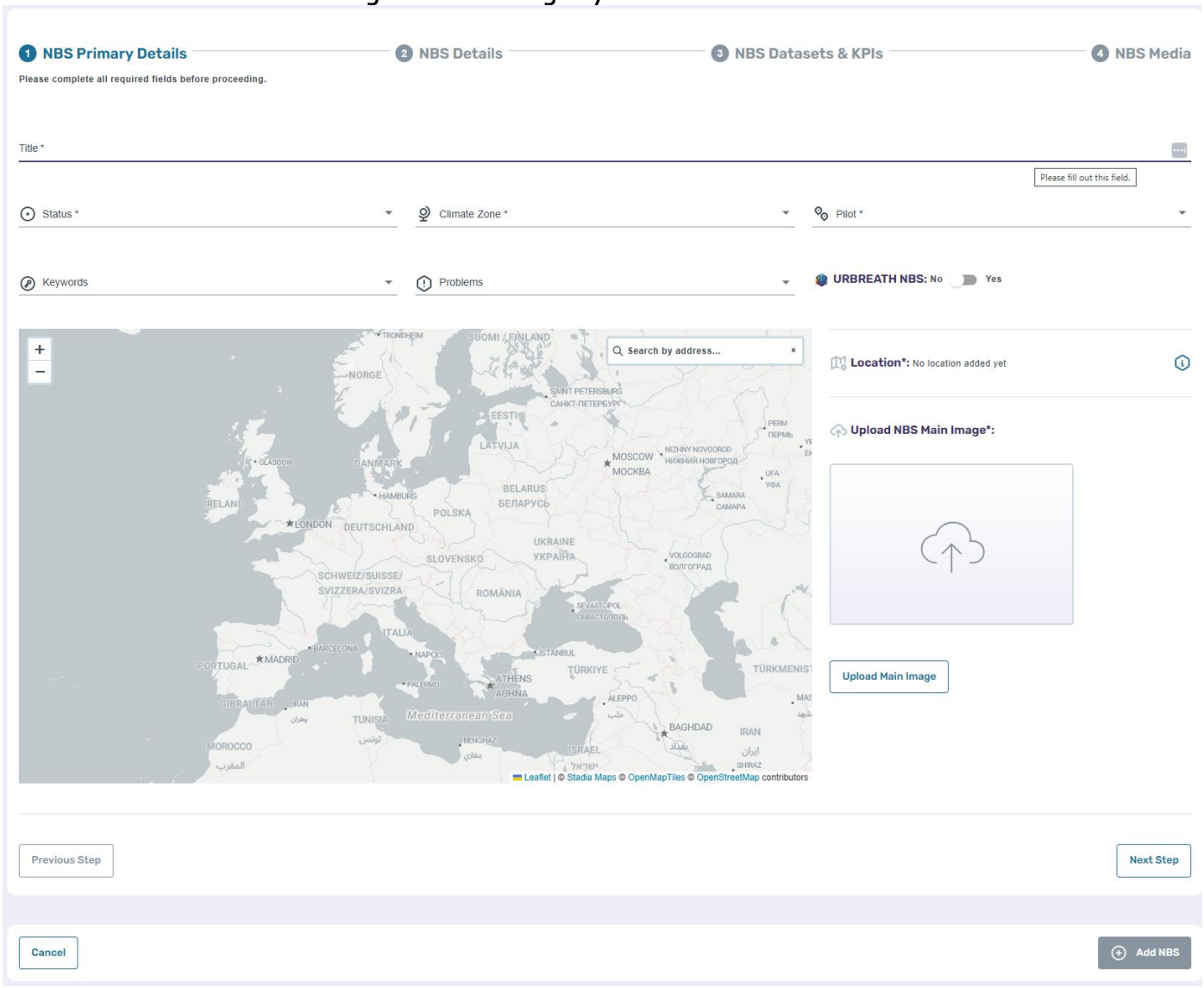


Figure 28: NBS Registry – Create new NBS



4.7.1.3 Technical implementation

The NBS Registry follows a 3-tier architecture as depicted in the diagram below.

In the data layer, the NBS Registry uses a PostgreSQL database to store the main information of the NBSs, a Minio server to store the actual media files (images and videos) uploaded by the users upon the NBS registration and a Redis database to enhance the performance of the application.

The business layer consists of three main subcomponents:

- The NBS Manager which offers APIs for the CRUD operations on the NBSs (GET, POST, PUT) and are also available to other tools within the URBREATH project like the Digital Twin.

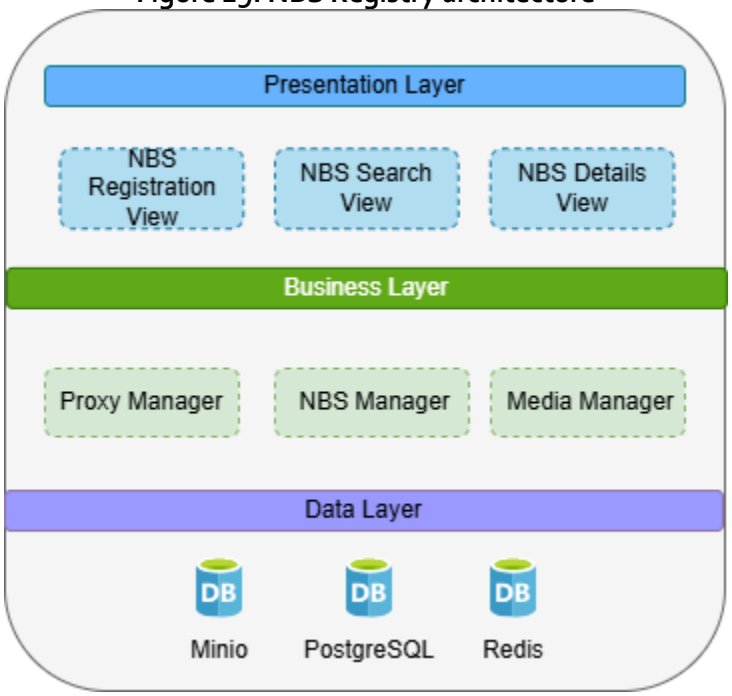
- The Proxy Manager which acts as the mediator between the Registry's UI and the URBREATH applications like the Dataset Catalogue and the KPI Manager.
- The Media Manager which provides an interface to store the media files (images and videos) in the Object Storage (i.e. MinIO, section 4.1.3).

The business layer is implemented using the Java Spring boot framework.

The presentation layer consists of the different NBS views and more precisely the registration view where a pilot administrator can add a new NBS, the search view where a user can search for NBSs and provides a map and box summary view of the NBSs and the details view where the user can get all the information related to a selected NBS.

The presentation layer is implemented using React JS and utilises components and techniques based on Material UI.

Figure 29: NBS Registry architecture



4.8 Data Discovery and Sharing

4.8.1 Datasets Catalogue, Models/Tools Catalogue, and Data sources Catalogue

4.8.1.1 Description and Purpose

The Dataset Catalogue, Model Catalogue, and Data sources Catalogue provide functionalities to search, discover, and identify information concerning:

- datasets managed by the URBREATH Toolbox (both generated by the toolbox itself, such as results from analysis, and offered by external systems, such as Open Data Management Systems, repositories, etc.)
- models and tools offered by the toolbox and/or managed by a Municipality (giving the opportunity to build a reference catalogue of tools and models), and
- data sources managed within a Municipality and/or external to a Municipality, but of interest (offering the chance to build also in this case a reference catalogue of useful data sources).

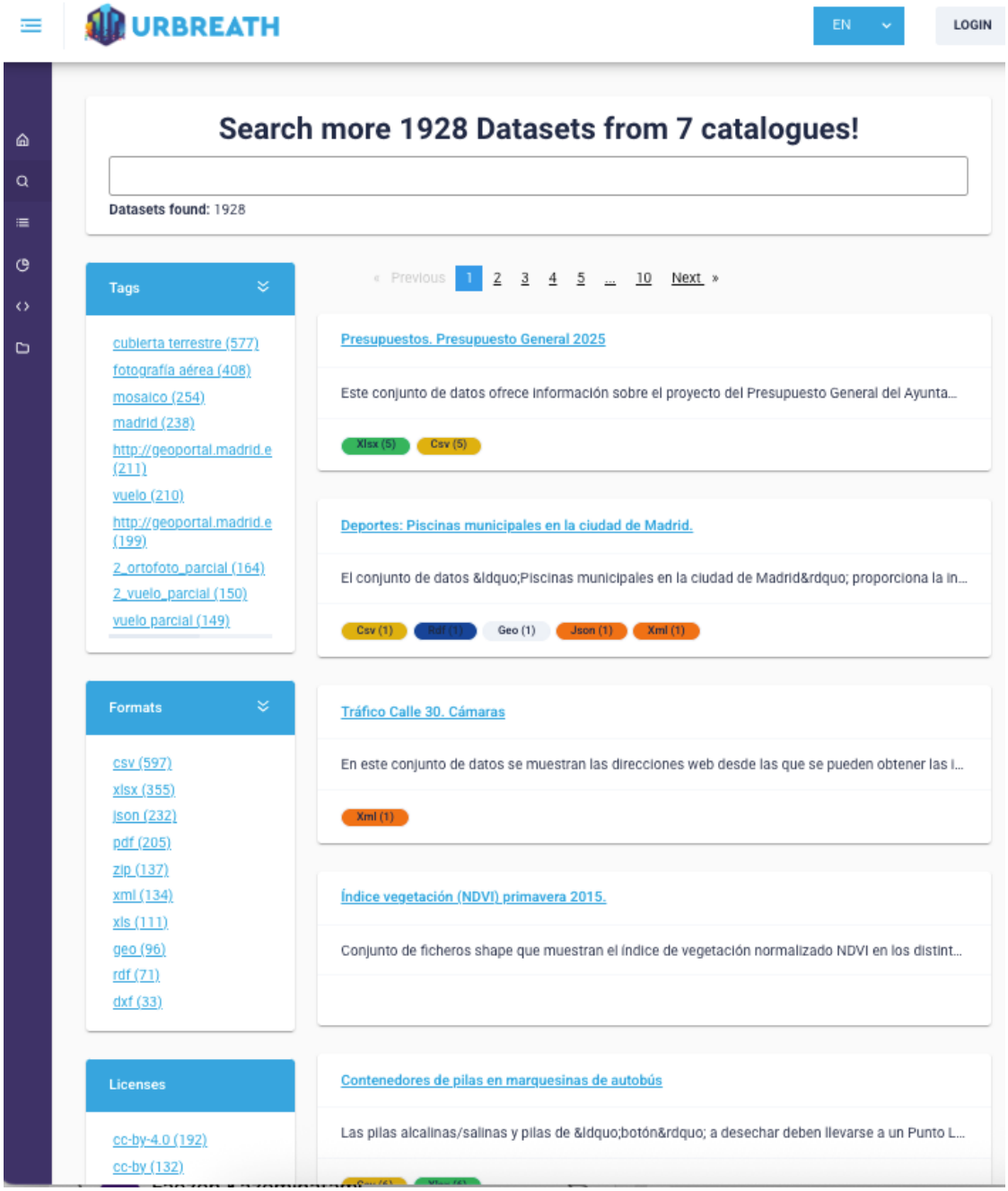
Each of the three catalogues offers a unified point to discover, identify and access information of interest, allowing the user to search datasets, models/tools, and data sources according to specific keywords and to filter the obtained results.

4.8.1.2 Key Features and Functionality

The three catalogues offer the same basic functionalities that are briefly summarised in the following.

Search by keywords. The user can input specific keywords to search among available datasets, models/tools, data sources. Results are displayed in a list; for each result the title and a preview of the description are reported (for the datasets, also the list of available data formats are reported).

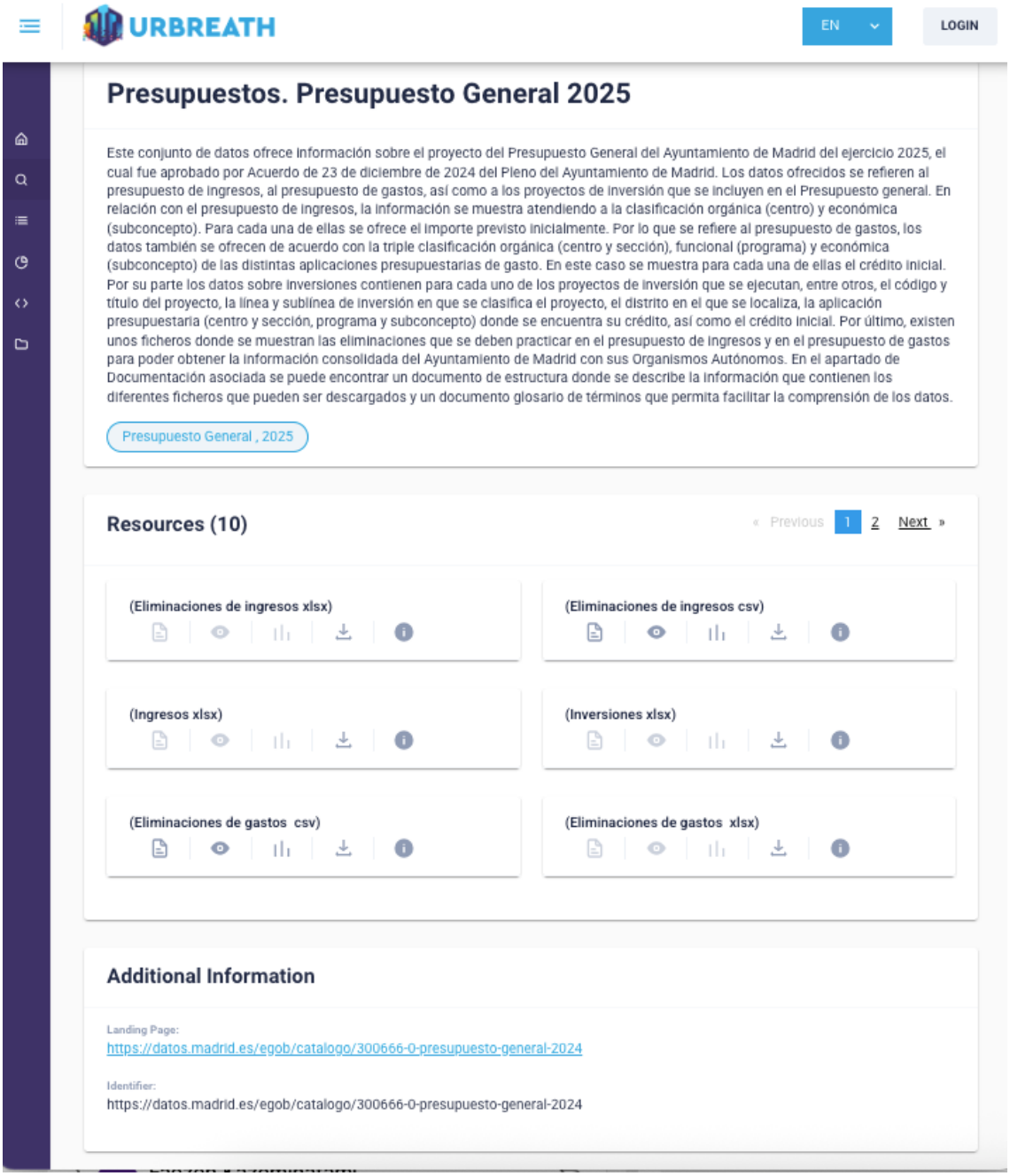
Figure 30: Datasets, Models/Tools, and Data sources Catalogues – Search UI



Filter results. The catalogues offer a list of filters that automatically identified according to the obtained results; the user can select one or more of the proposed filters to refine obtained results. Offered filters can be, for instance, main tags associated to the results, license, data formats, etc.

Access details of the results. By clicking on the title of a result, the user can access its details. Details of the results include the full description, the full list of tags, and the associated resources (e.g. files associated with a dataset, URL of the source code of a tool, end point of a data source, etc.)

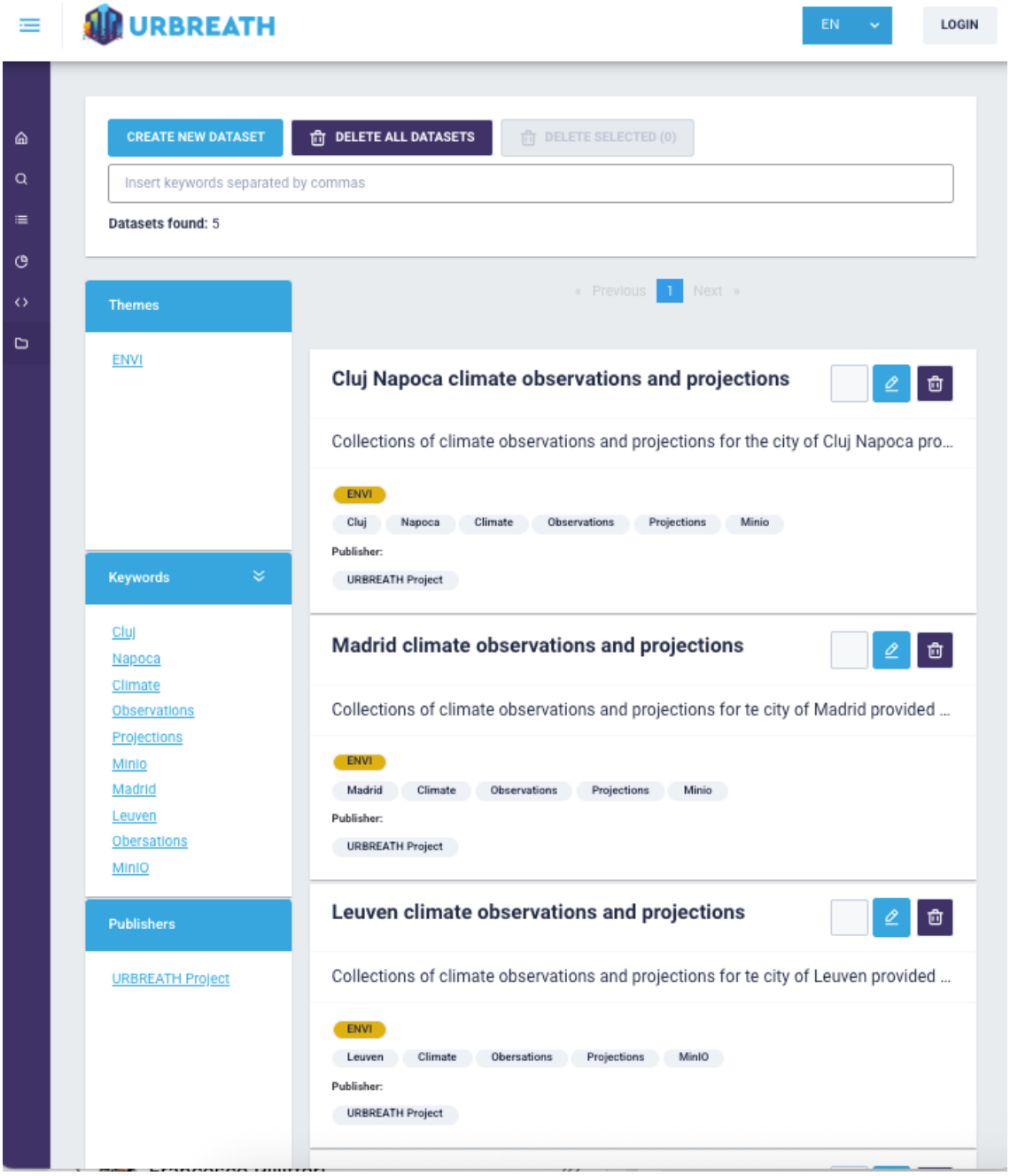
Figure 31: Datasets, Models/Tools, and Data sources Catalogues – Details of results



Management of datasets, models/tools, and data sources. The user can manage his or her datasets, models/tools, and data sources. The catalogues offer specific sections, where the user can access the

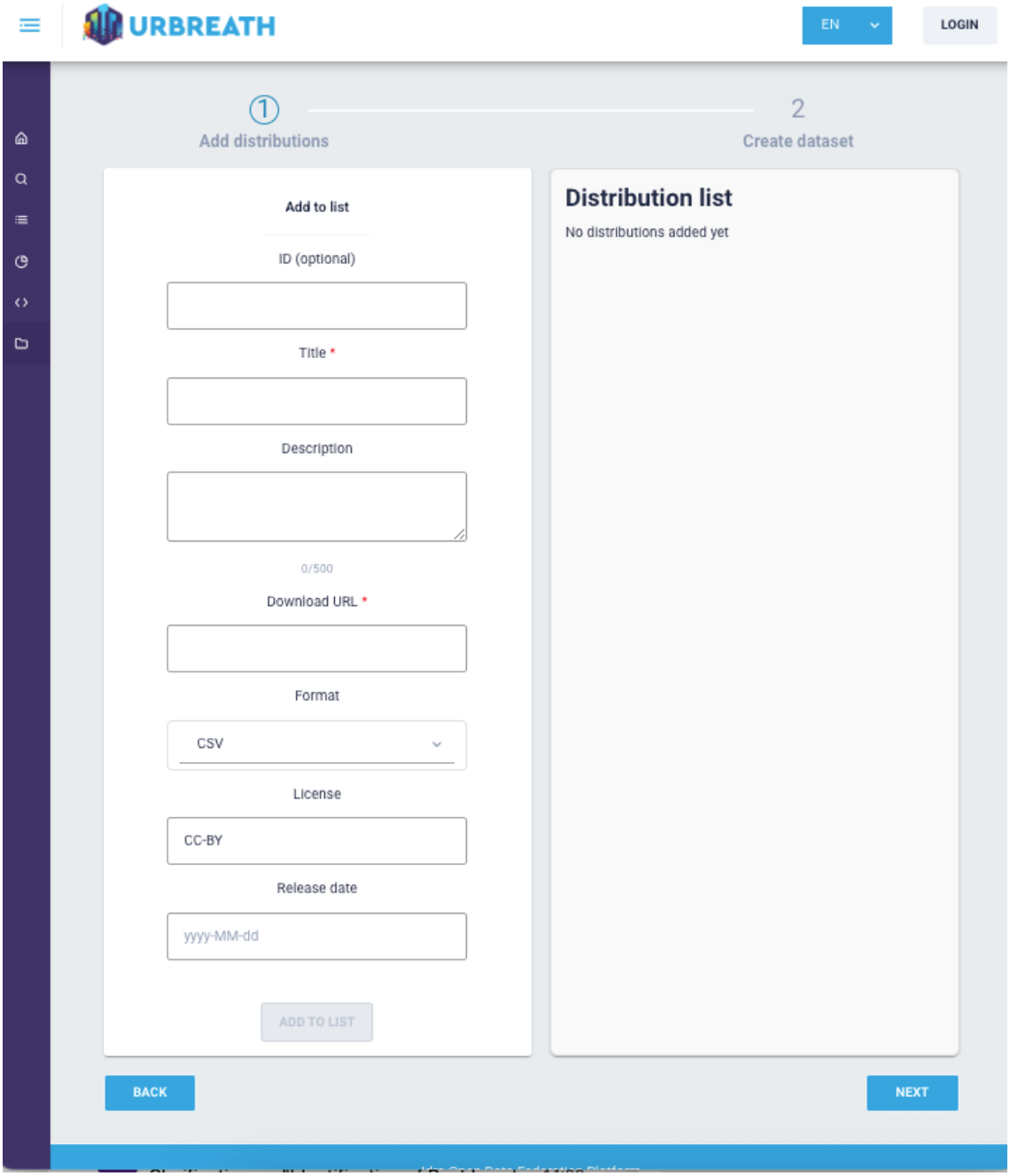
list of datasets, models/tools, and data source he/she created, create a new one, edit/delete existing ones.

Figure 32: Datasets, Models/Tools, and Data sources Catalogues – Management



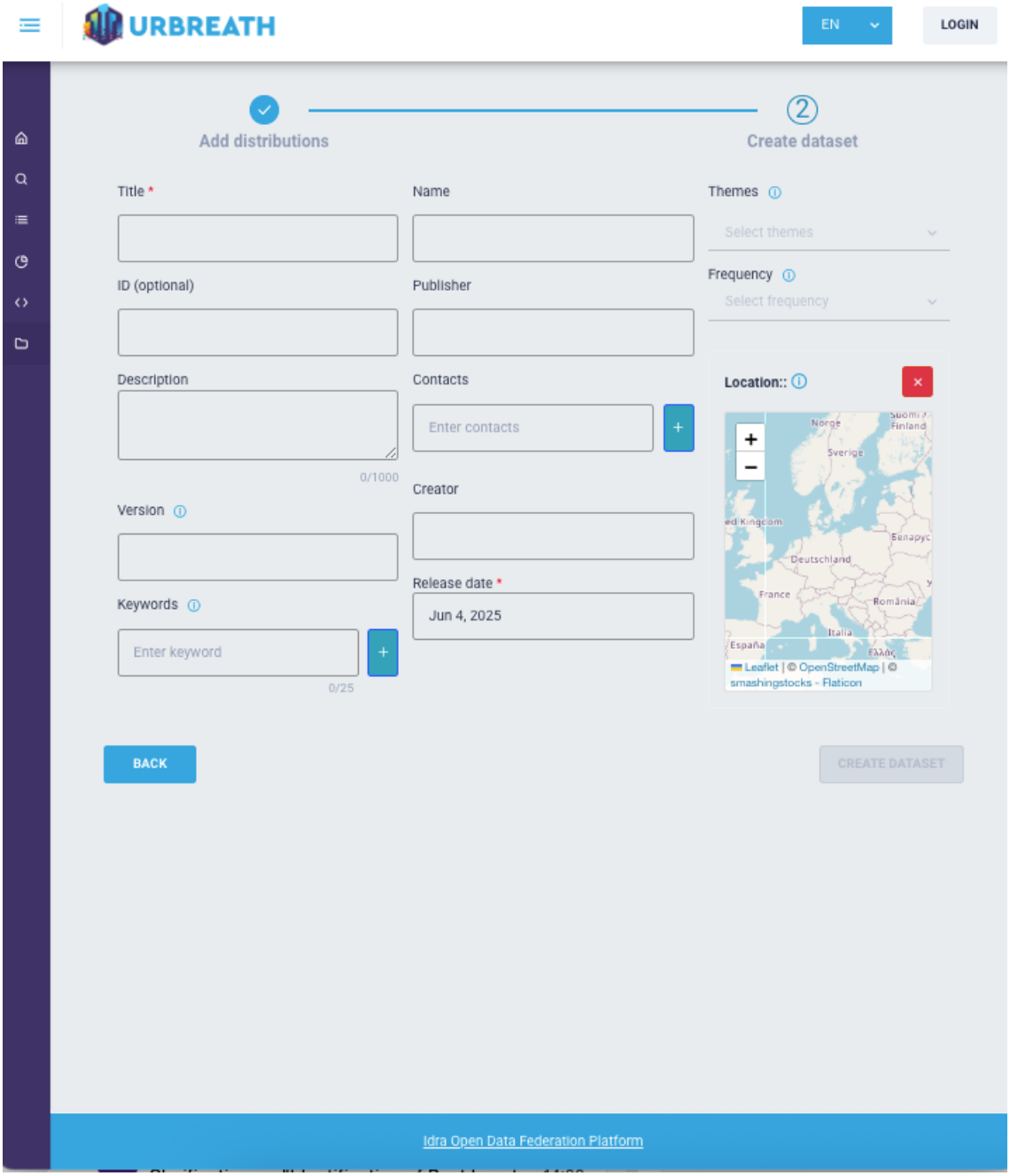
When the user starts the creation of a new dataset, model/tool, or data source, the catalogue displays a wizard. In the first step, the user adds information related to the resources associated with a dataset, model/tool, or data source.

Figure 33: Datasets, Models/Tools, and Data sources Catalogues – Add a new resource



In the second step, the user adds general information about the new dataset, model/tool, or data source, such as the title, the description, the version, keywords, geographic location, etc.

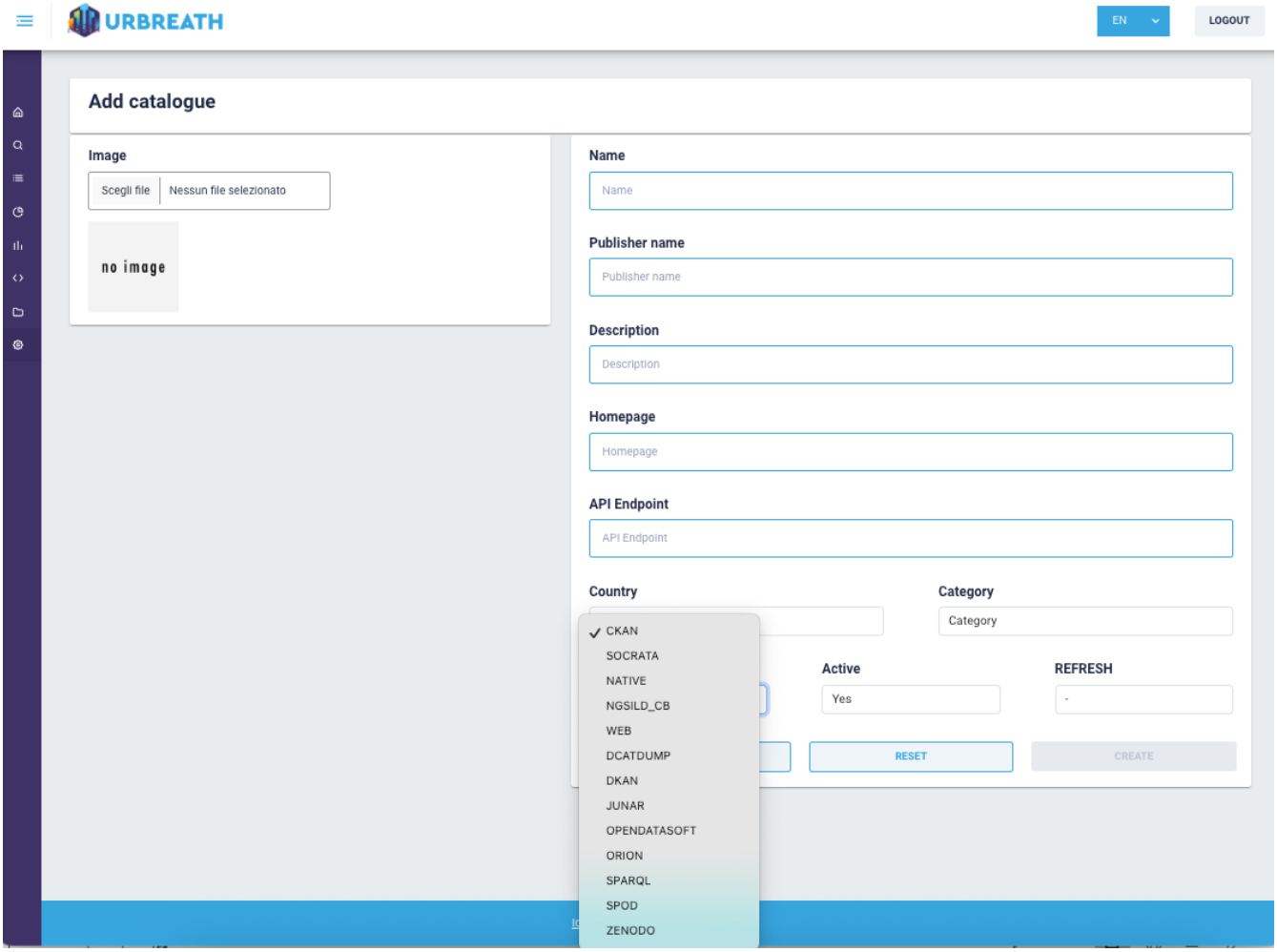
Figure 34: Datasets, Models/Tools, and Data sources Catalogues – Add general information



The publication of information about new datasets, models/tools, and data sources, is possible thanks to the integration with the Publisher component of the URBREATH Toolbox (section 4.8.3)

Federation of third-party catalogues of datasets. In addition to the functionalities reported above, the Dataset Catalogue can federate catalogues managed by third parties. For this purpose, the Dataset Catalogue supports different technologies such as CKAN, SOCRATA, DKAN, Orion Context Broker, SPOD, ZENODO, etc., thanks to Data Catalogues Connectors (section 4.2.1)

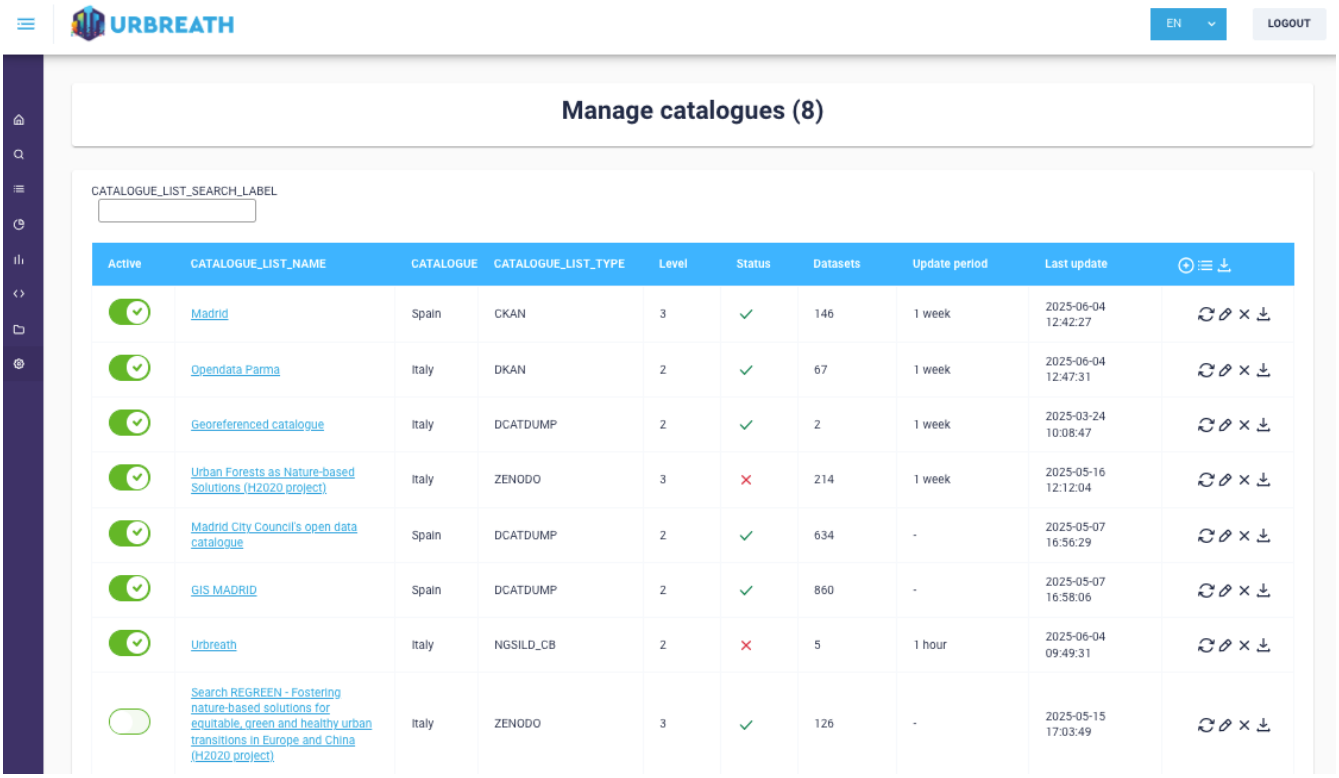
Figure 35: Datasets Catalogue – Federate an external catalogue



To federate a catalogue, the user adds its details (e.g. title, description, etc.), specifies its endpoint, the refresh period (i.e. the time interval after which the catalogue checks for updates), and marks it as active or not active. If active, the federated catalogue is made available and the datasets it provided are reported among results provided to the users; if not active, the federated catalogue is in a "stand by" mode. The user can change the status (active vs not active) in a second moment.

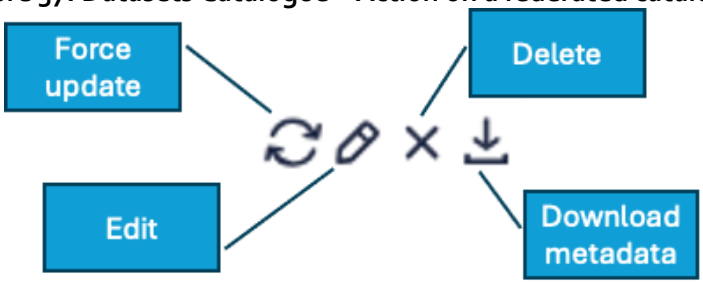
All the federated catalogues are reported in a dedicated section, from which the user can manage them and access relevant information, such as the technology on which the federated catalogue is based, the update period, the date and time of the last update.

Figure 36: Datasets Catalogue – Management of federated catalogues



In addition, by action on dedicated buttons, the user can force the update of the federated catalogue, edit the federated catalogue, delete the catalogue, and download the entire set of metadata of the federated catalogue in DACT-AP format.

Figure 37: Datasets Catalogue – Action on a federated catalogue



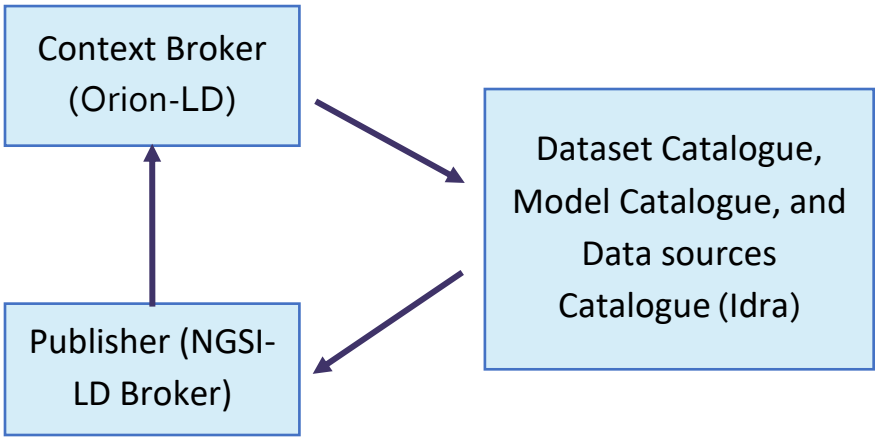
4.8.1.3 Technical implementation

From a technical perspective the Dataset Catalogue, Model Catalogue, and Data Sources Catalogue are based on Idra, an open-source web application designed to federate scattered and heterogeneous catalogues of data under a unique point of access, allowing the end users to search and discover datasets they provide. To this aim, Idra offers a series of connectors able to interact with a wide range of technologies for managing catalogues of data, such as CKAN, DKAN, JUNAR, OPENDATASOFT, ORION-LD Context Broker (leveraging DACT-AP entities NGSI-LD compliant), SOCRATA, and SPOD. Idra collects metadata from federated catalogues, offering a uniform representation of collected metadata according to DACT-AP. This allows Idra to offer the possibility to the end users to search

across different catalogues at the same time. Detailed information can be accessed on the official documentation[28]. Search capabilities of Idra are accessible from a dedicated Web UI and REST APIs[29].

In URBREATH, the user interface of Idra has been extended to include new the sections for the management of user created entries concerning datasets, models/tools, and data sources (briefly described in section 4.9.2.2). Idra has been also integrated with the Publisher component of the URBREATH Toolbox, to allows the concreate management of those user-created entries, making them available for future searches.

Figure 38: Datasets, Models/Tools, and Data sources Catalogues – Components for managing new resources



The new sections of the UI for managing user created entries concerning datasets, models/tools, and data sources interacts with the NGSI-LD Broker (part of the Publisher component of the URBREATH Toolbox), that offer the possibility to perform CRUD[30] Operations on NGSI-LD entities following the DCAT-AP format. These NGSI-LD entities are stored into the Context Broker component of the URBREATH Toolbox, that is based on the ORION-LD Context Broker (section 4.3.2). The Context Broker is then federated into Idra leveraging its ORION-LD Context Broker connector, that allows to expose the managed DCAT-AP NGSI-LD entities to the end users for search purposes.

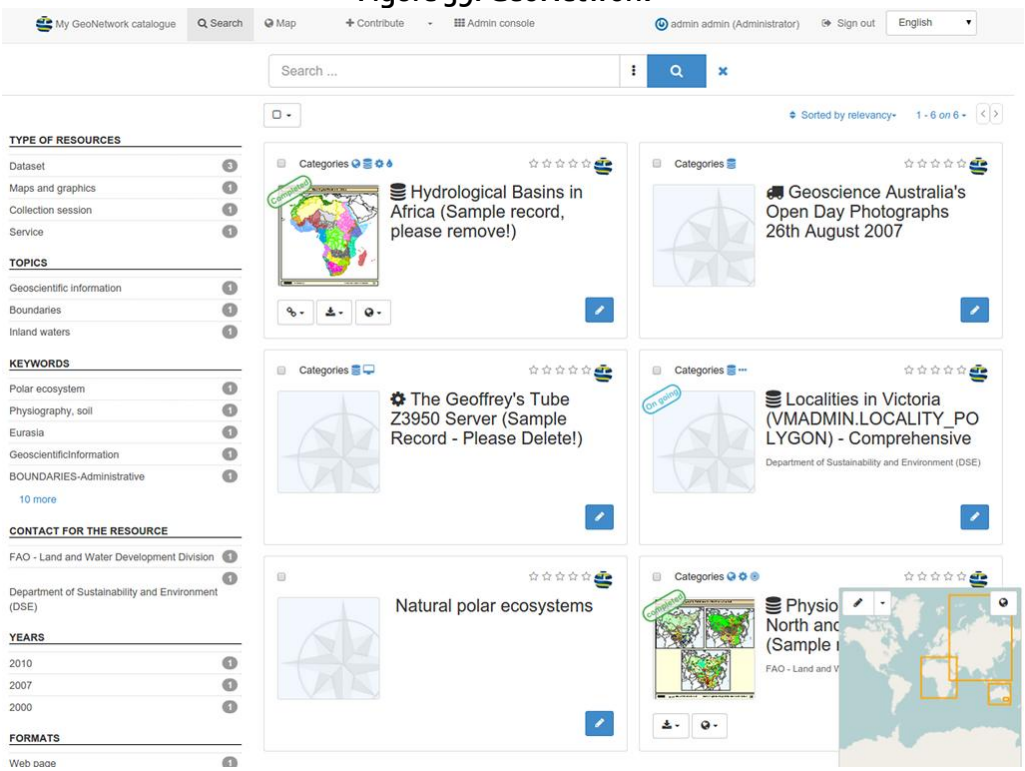
4.8.2 GeoNetwork

4.8.2.1 Description and Purpose

GeoNetwork is an open-source catalogue application designed to manage and share spatial datasets and metadata. It helps users organize, discover, and access geospatial resources through a web interface with powerful search and visualization tools. Built on international standards like ISO and OGC, it supports metadata editing, harvesting, and interoperability.

[28] https://idra.readthedocs.io/en/latest/
[29] https://idraopendata.docs.apiary.io/#reference/end-user-api
[30] Create Read Update Delete

Figure 39: GeoNetwork[31]



4.8.2.2 Key Features and Functionality

GeoNetwork key functionalities include:

- Fast access to local geospatial catalogues.
- Automated harvesting and synchronization between distributed catalogues using multiple protocols (GeoNetwork, CSW, OGC WxS GetCapabilities, WebDav, ArcSDE, Thredds, OGC WFS Features, OAI-PMH).
- Native support for ISO and Dublin Core metadata standards.
- Uploading and downloading of various content types (tables, documents, images).
- An interactive Web Map Viewer to combine WMS layers from multiple servers.
- Online metadata editing with customizable templates.
- Fine-grained access control and comprehensive user/group management.
- Multi-lingual interface for global accessibility.

[31] Source: https://live.osgeo.org/it/overview/geonetwork_overview.html

4.8.2.3 Technical implementation

GeoNetwork is developed in Java and runs on standard servlet containers like Apache Tomcat. The backend relies on a relational database (commonly PostgreSQL/PostGIS) for storing metadata and spatial information, enabling advanced geospatial queries and indexing.

The system supports OGC standards such as CSW (Catalog Service for the Web) for metadata harvesting and search and integrates seamlessly with various spatial data infrastructures. GeoNetwork's modular architecture allows easy extension and customization through plugins and templates. It also provides RESTful APIs to facilitate integration with external systems and automated workflows.

4.8.3 Publisher

The Publisher component of the URBREATH Toolbox logical architecture offers capabilities to publish within the toolbox content of interest such as information related to datasets, software models/tools, data sources and 3D models (e.g. to be visualised into the 3D Map component).

From a technical perspective, this logical component includes two distinct software tools:

- the NGSI-LD Broker, which is employed for the publication of the metadata related to datasets, models/tools, and data sources and
- the VC Publisher as an optional tool, which allows to publish data sources and 3D Maps, by supporting continuous updates and of the digital representations of real-world assets.

The NGSI-LD Broker offers a set of REST APIs that allows to perform CRUD operations on NGSI-LD compliant entities representing DACT-AP datasets.

The provided APIs accept as input JSON-formatted data representing the datasets to be managed and transform input data into the corresponding NGSI-LD entities (https://github.com/smart-data-models/dataModel.DCAT-AP). These NGSI-LD entities are then stored in the Context Broker (also part of the URBREATH Toolbox; see section 4.3.2), which will manage their life cycle.

To fully represent a dataset, the NGSI-LD Broker generated two types of NGSI-LD entities:

- Dataset: includes the general information of a dataset, such as title, description, keywords, etc.
- Distribution: includes information related to a concrete resource associate to a dataset (e.g. a file), such as title, description, download URL, file format, etc.

A dataset entity can be associated to multiple distributions.

The APIs offered by the NGSI-LD Broker allow to perform the following actions on both Datasets and Distributions:

- Create a new Distribution: add a distribution into the Context Broker.
- Create a new Dataset: add a dataset into the Context Broker.

- Update a Distribution: update a distribution that is already registered into the Context Broker.
- Update a Dataset: update a dataset that is already registered into the Context Broker.
- Delete a Distribution: delete a distribution that is already registered into the Context Broker.
- Delete a Dataset: delete a dataset that is already registered into the Context Broker.
- Retrieve All Distributions: returns all the distributions saved into the Context Broker.
- Retrieve All Datasets: returns all the datasets saved into the Context Broker.

These APIs employed in the Dataset Catalogue, Model Catalogue, and Data Sources Catalogue for the management of related information from their UI (see section 4.9.2)

On the other hand, the **VC Publisher** is a powerful "multitool" designed for creating and publishing 3D web maps and digital city models. It takes diverse geospatial inputs—such as terrain models, aerial imagery, point clouds, vector data, and CityGML—and transforms them into high-performance, web-optimized visualization datasets (Cesium streaming data). It can be installed on-premises or deployed in the cloud and includes automation and plugin support for scalability and customization. It integrates seamlessly with other modules like:

- VC Warehouse (for data conversion and exchange)
- VC Planner (for planning workflows)
- VC Database (for CityGML data administration and querying)

Key Capabilities of VC Publisher

1. Efficient Data Conversion
    a. Converts raster, vector, CityGML, orthophotos, terrain, point clouds, and oblique aerial images into web formats.
    b. Highly performant converters accelerate processing and publishing
2. Automated Publishing
    a. Publishes 3D web maps via VC Map in one click.
    b. Supports scheduled, automated workflows—keeping data and apps up to date after each update
3. Plugin Support & Customization
    a. Offers plugin-based architecture enabling developers to add custom data processing or tools.
    b. Full API extensibility for tailored workflows.
4. Web Standards & OGC Services
    a. Integrates OGC WMS, WMTS, WFS services seamlessly.
    b. Features XML validation and coordinate system transformations for robust data input
5. Flexible Deployment
    a. Deployable on local servers or in the cloud, suiting various IT environments
6. Integration with VCS Ecosystem
    a. Works together with VC Warehouse, VC Planner, VC Database, and VC Map—forming a full-stack solution from data ingestion to visualization

The VC publisher exposes a REST API and a Web UI for management purposes.

4.8.3.1 Overview of the VC Publisher REST API

The API enables:

- Authentication via token-based or API-key mechanisms
- File operations, including upload/download of geodata packages
- Task management, such as creating, launching, monitoring, and deleting processing jobs
- Integration hooks for tools like FME Server, enabling chained workflows and automation
- In total, there are over 120 REST endpoints and more than 20 custom transformers (for FME) described in the OpenAPI spec, covering nearly every aspect of VC Publisher's functionality

What You Can Achieve with the REST API

Automated Data Ingestion

Upload aerial imagery, point clouds, CityGML, terrain files, orthophotos, etc., directly via API.

On-Demand Conversion & Publishing

Trigger conversion and Cesium 3D tile generation on a schedule or on demand.

Chained Workflows & Integration

Use external tools (like FME Server) to upload → convert → publish all in one API-driven pipeline, keeping maps up-to-date automatically

Job Management & Monitoring

Programmatically start tasks (e.g., "Process this dataset"), check progress, cancel jobs, and handle output download.

Seamless Integration

With the OpenAPI spec, you can import the REST definition directly into tools like FME (since FME 2023.1) or Postman, facilitating integrations

Scalability & Orchestration

Chain multiple tasks (e.g. "Upload → Convert → Publish"), enabling fully automated, repeatable pipelines.

4.9 City Data Visualisation

4.9.1 Unified UI

4.9.1.1 Description and Purpose

The **Unified UI** is a user interface element that serves as the **wrapper** for other URBREATH UI components. Its primary purpose is to organize content, enforce layout rules, and provide styling or interactive context to the elements it contains. The Unified UI sets the design guidelines that all the rest of the embedded UIs must follow to achieve a common look n' feel.

4.9.1.2 Key Features and Functionality

The Unified UI provides container UI with common elements like the header, menu, footer, etc. and it enables basic user functionalities like login, user management and user settings administration, as well as an introductory page for the NBS stories. The Unified UI passes data to the embedded UIs like the user token that enables these UIs to present the user with the proper views, e.g. based on the user's role.

The UI is organized in 3 main sections:

- The header on the top contains buttons for changing the appearance of the page like having the menu collapse or view in full screen, as well as allowing the user to login.
- The main menu on the left allows the user to investigate the various tools of the URBREATH system like the Data Catalogue, the NBS Registry etc., as well as view the most important NBS stories organized by climate zone.
- The central part of the page is where the user can utilize the functionalities of the project's tools and view information.

The user initially views the landing page of the Unified UI, which includes brief information about the project like what we offer from a technical point of view, why someone to choose our platform and a few statistics regarding the NBSs, the climate zones and pilots we cover (Figure 40). Depending on what the user selects on the menu, the relevant UI appears, for example the NBS Stories are depicted in Figure 41.

Figure 40: Unified UI

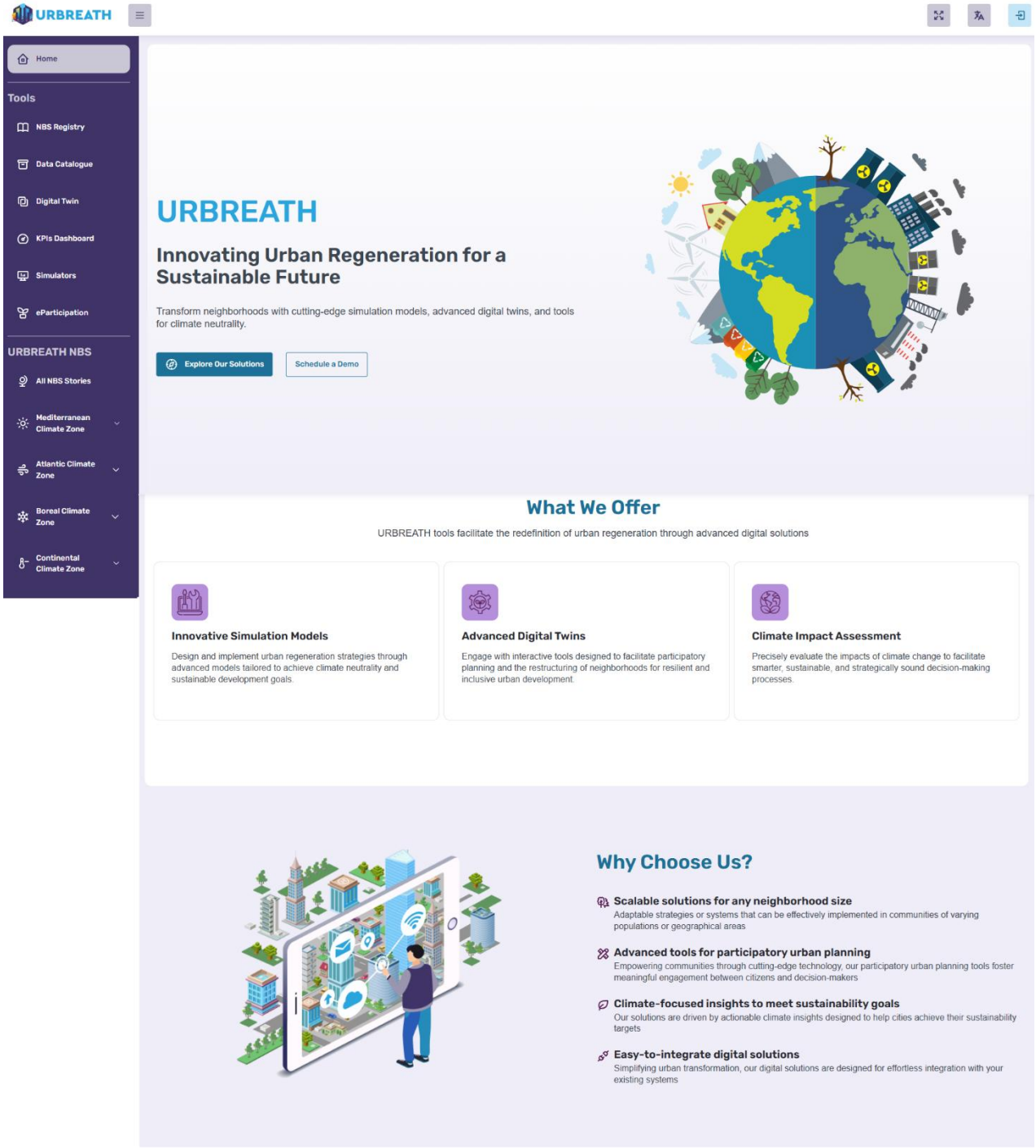
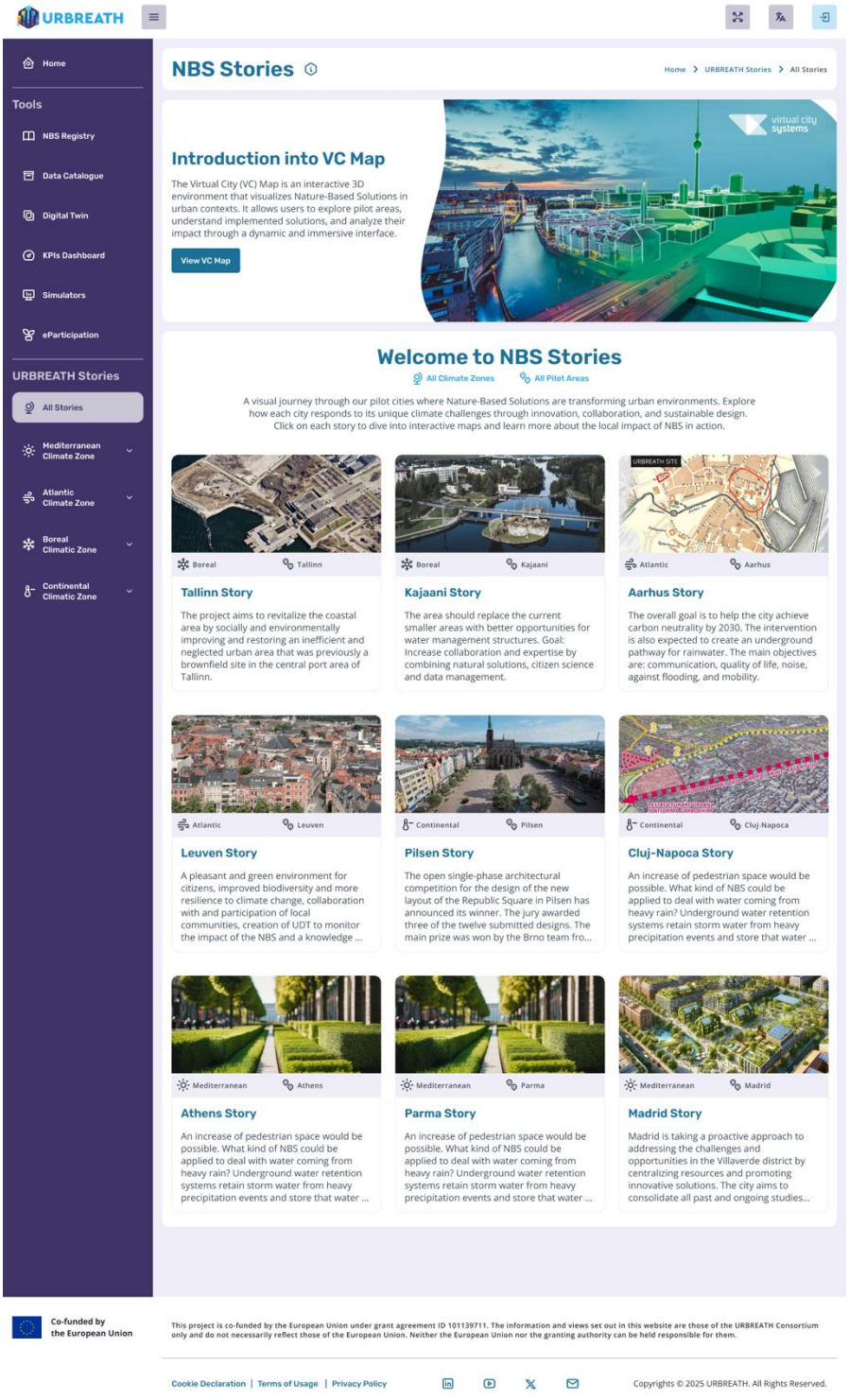


Figure 41: NBS Stories



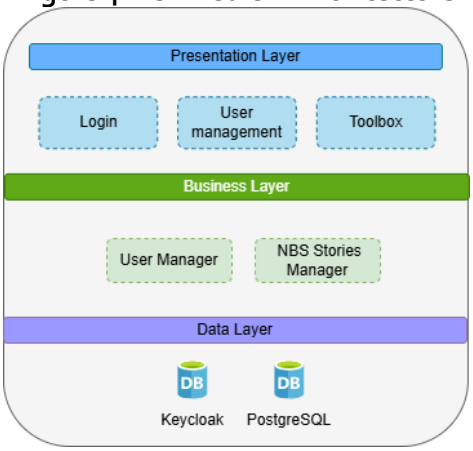
4.9.1.3 Technical implementation

The Unified UI's internal architecture is depicted in the following diagram. Although primarily a UI component, it is backed up by backend services and subsequently by a database.

The presentation layer is comprised of the Login view where the users have to enter their credentials in order to enter the project's toolbox, the user management view which is visible only to administrators and through which the users of the system can be created, updated or deleted and finally the toolbox view where all the URBREATH UIs are embedded and presented to the user in a unified view. The presentation layer is implemented using React JS and utilises components and techniques based on Material UI.

The business layer is comprised of the User Manager which is a component acting as the mediator between the UI and the Identity Manager (Keycloak) and the NBS stories Manager which fetches the NBS stories from the database. The business logic layer is implemented using Java Spring Boot.

Figure 42: Unified UI - Architecture



4.9.2 Data Visualisations and Dashboards

4.9.2.1 Description and Purpose

The KPI dashboards developed by Municipia are designed as advanced, web-based decision-support tools to monitor, analyse, and visualize data from urban areas involved in Nature-Based Solutions (NBS) interventions. Leveraging the capabilities of Apache Superset, these dashboards integrate data from IoT sensors, open-source datasets, and project-generated indicators to track and visualize Key Performance Indicators (KPIs) over time and space.

Their primary objective is to provide city officials, urban planners, and technical partners with real-time and historical views of urban metrics relevant to climate neutrality, resilience, and regeneration goals. The system enables cities to assess whether current performance aligns with predefined target values and supports informed decision-making and strategic planning through a comprehensive and interactive visual interface.

4.9.2.2 Key Features and Functionality

Dashboard Management (Apache Superset-powered)

- Creation, editing, and deletion of dashboards via a user-friendly, web-based interface.
- No-code UI for quick chart and dashboard design through drag-and-drop and point-and-click functionality.
- Web-based SQL Editor for advanced users to write and execute custom queries on connected data sources.
- Role-based access control (RBAC) to manage data visibility and user privileges.

Visualisation Types

- Superset provides a wide range of built-in visualizations for rich data representation, including:
- Line charts, bar charts, pie charts
- Time-series graphs
- Geospatial visualizations (map-based layers using GeoJSON or WMS services)
- Heatmaps, box plots, KPI cards, and gauge indicators
- Combined plots and multi-chart dashboards

KPI Integration

- Dashboards include dynamic KPI tracking widgets, enabling the visual monitoring of specific indicators (e.g. air quality index, average pedestrian flow, green surface area coverage).
- KPIs are sourced from real-time sensor data, open datasets, and manual evaluation processes (e.g., surveys, participatory assessments).
- Superset allows aggregation, threshold configuration, and conditional formatting for KPI panels to signal deviations or alerts.

Historical Data Integration

- Historical time-series data is imported into the underlying data warehouse (e.g., PostgreSQL, MySQL), enabling users to visualize trends and patterns over time.
- Dashboards support temporal filters and date-range selectors for exploratory analysis and KPI evolution tracking.

Interactive and Comparative Analysis

- Custom filtering and drilldowns by sensor, geography, time, or scenario.
- Possibility to compare performance across different NBS areas or between pilot cities.
- Exportable dashboards and individual chart elements for reuse in presentations and external documentation.

4.9.2.3 Technical implementation

The KPI dashboard infrastructure developed by Municipia is built on Apache Superset, a robust, pen-source business intelligence and data visualization platform optimized for handling large-scale analytics in web environments. The platform is deployed within a containerized microservice architecture—typically using Docker—to ensure modularity, portability, and efficient orchestration across environments. Data ingestion, transformation, storage, and querying are all managed within a relational database management system (RDBMS) such as PostgreSQL, which serves as the unified data backend for the dashboards.

Data sources integrated into the platform include both streamed IoT sensor data and open-access datasets. IoT data is collected from urban areas undergoing NBS (Nature-Based Solution) interventions, via sensor networks that monitor different variables for example particulate matter ($PM_{2.5}$), nitrogen dioxide (NO_2), CO_2, temperature, humidity, and related climatic indicators. These data streams are ingested through ETL pipelines that handle preprocessing tasks including data normalization, timestamp synchronization, and missing data imputation. Additional data is obtained from municipal open data portals and partner-provided datasets, following schema alignment and validation procedures to ensure consistency across heterogeneous sources.

Superset connects to these structured datasets via SQLAlchemy[32], offering a high degree of compatibility and extensibility. Once the data is available, Superset's semantic modelling layer is used to define logical groupings, custom metrics, calculated fields, and aggregate functions. These abstractions serve as the foundation for constructing reusable KPI components and domain-specific data views. Superset's support for time-series analysis is crucial here, as it enables historical trend tracking, real-time updates, and temporal filtering for monitoring performance indicators over selected intervals.

To facilitate both technical and non-technical user engagement, the dashboard interface supports two primary modes of interaction. The first is a visual no-code editor, allowing users to build dashboards through drag-and-drop operations, parameterized filters, and modular widgets. The second is a web-based SQL editor, designed for advanced users who require complex queries, joins, or subqueries across relational datasets. Both interfaces are integrated with Superset's visualization engine, which offers a rich library of data representation options—such as bar charts, line graphs, area charts, KPI cards, pie charts, and geospatial layers rendered through Mapbox[33] or Leaflet integrations (e.g., via GeoJSON or WMS feeds).

Currently, a demonstration prototype of the dashboard has been developed, focusing on two priority KPI domains: air quality and climate temperature. These KPIs are computed from live and historical sensor data and visualized using time-series plots, threshold-based indicators, and map-based

[32] A Python toolkit that simplifies the management of relational databases and related queries; https://www.sqlalchemy.org/

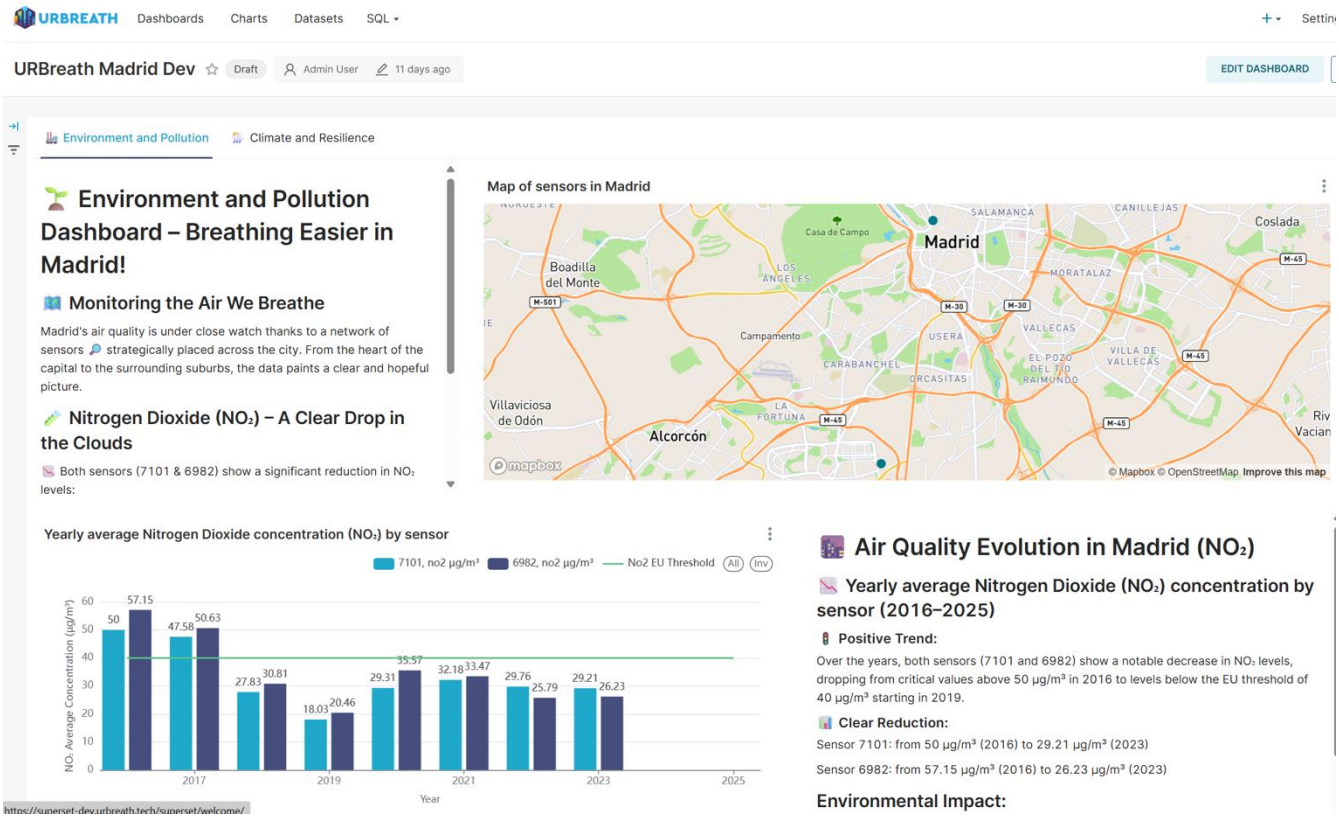
[33] A cloud platform providing software tools to implement map-based capabilities (e.g. navigation) into software applications (e.g. mobile applications); https://www.mapbox.com/

distributions. The demo serves as a proof of concept for the system's ability to process environmental data streams, calculate KPI values against predefined targets, and render multi-scale visual outputs for urban stakeholders.

Security and access control are managed through Superset's role-based access control (RBAC) framework, which ensures that user permissions can be finely tuned by dataset, dashboard, and feature scope. This guarantees controlled access to sensitive or city-specific data layers. Moreover, the platform supports report generation, enabling users to export full dashboards or individual visualizations in PDF format, which can be embedded into presentations, evaluation reports, or shared among stakeholders. Future iterations may include scheduled report automation and integration with notification systems to distribute KPI updates on a recurring basis.

In summary, the technical implementation of the Municipia dashboard suite offers a scalable, secure, and extensible analytics environment capable of real-time and longitudinal monitoring of urban sustainability indicators. It leverages a modern data stack with open-source tools to enable data-driven decision-making across pilot cities and is designed for replicability, modular expansion, and alignment with broader smart city and climate neutrality strategies.

Figure 43: Data Visualisations and Dashboards

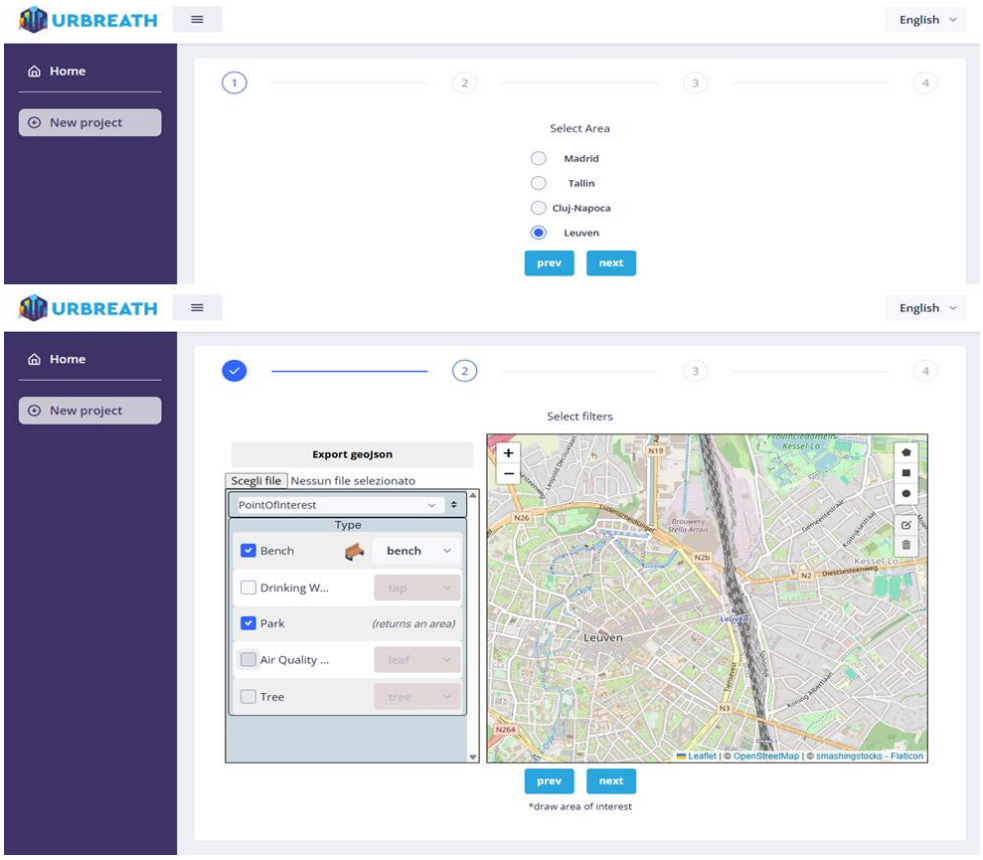


4.9.3 Map Layers Manager

4.9.3.1 Description and Purpose

The Map Layers Manager is based on the GeoCacher, a web application designed to simplify the management and publication of geographic data in GeoJSON format. The main goal of this application is to enable various users to easily explore and visualize geospatial data and generated customised map layers as GeoJSON. The GeoCacher interfaces with the ORION context broker, which is the primary source to retrieve geo referenced information derived from NGSI-LD entities. It is built with Java (Spring Boot), leverages Docker for containerization, and includes supporting databases like MongoDB.

Figure 44: Map Layers Manager



In practical terms, GeoCacher provides a straightforward way to transform user-uploaded GeoJSON files into interactive map layers, accessible through standard web services.

4.9.3.2 Key Features and Functionality

The GeoCacher offers to not skilled users an easy-to-use UI to explore geo referenced information, extract information of interest and generate map layers that can also be published as datasets

(through the integration with the Publisher components of the "Data Access and Sharing" macro area, and/or on the GeoServer.

The user is guided in exploring available geo referenced information with a wizard, that proposes a series of filters through which selected information of interest. The proposed filters are dynamically generated according to the available geo referenced information.

The user is provided with a map on which he/she can draw one or more areas on which performs the geographic search according to the selected filter. The user can draw multiple squares, circles, or custom polygons.

Once the search criteria have been defined (i.e. filters and areas of interest), the user can visualise the obtained results and change the drawn areas and filter or save the generated map layers. In this case, the user inserts a title, a description, and can select the possibility to auto update the map layer at regular intervals (e.g. every day, every week, etc.).

Finally, the user can publish the defined map layers as datasets (on the Dataset Catalogue) or export it on the GeoServer, to make it accessible for future use.

4.9.3.3 Technical implementation

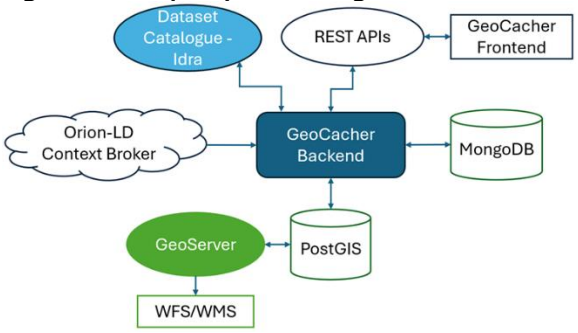
The Map Layer Manager is implemented on top of GeoCacher, originally developed as a modular Spring Boot microservice for managing user-defined spatial data. The backend relies on MongoDB; all data (including user-defined filters, GeoJSON geometries and metadata) persisted as flexible, schema-less JSON documents across different collections.

This architecture supports dynamic data modelling, making it easy to evolve the data structure without complex migrations. MongoDB's native geospatial indexing allows efficient spatial filtering, enabling users to interactively define areas of interest (e.g., polygons, circles, multi-polygons) and retrieve relevant geographic features based on spatial and semantic constraints.

The system exposes a REST API layer that supports full CRUD operations for layers and jobs. A Quartz-based job scheduler is used to automate the periodic execution of saved filters and regeneration of associated layers. These jobs are configured by users during the layer creation process, enabling real-time synchronization with context changes in the city.

Additionally, the Map Layer Manager is integrated with the Dataset Catalogue component (Idra), allowing users to publish generated layers as discoverable datasets. This integration ensures that spatial information becomes part of the federated data ecosystem, supporting search, discovery, and reuse in alignment with DCAT-AP standards.

Figure 45: Map Layers Manager – Architecture



To support standardized geospatial publishing, the Map Layer Manager integrates with GeoServer and uses PostGIS as the spatial database backend. When a user marks a layer for publication, the system takes the GeoJSON data and performs two actions:

- It stores the spatial data into PostGIS, creating a dedicated table that contains both geometry and associated attributes.
- It then uses the GeoServer REST API to create a corresponding workspace, datastore, and layer, referencing the newly created PostGIS table.

This setup allows the published layers to be accessed through standard OGC protocols such as WMS (for map rendering) and WFS (for structured data access).

Overall, the Data Layer Manager bridges flexible, user-driven geospatial data interaction with standardized geodata publication pipelines, offering a powerful backend service to support smart city dashboards, spatial analytics, and open data portals.

4.9.4 3D Map

4.9.4.1 Description and Purpose

The 3D Map component of the URBREATH Toolbox is based on **VC Map**. Developed by Virtual City Systems, VC Map is a versatile web-based platform designed for visualizing, analysing, and sharing 3D city models and geospatial data. VC Map is an open-source platform designed for visualizing 2D and 3D geospatial data on the web, providing a powerful yet accessible tool for creating interactive maps and local digital twins. Built on CesiumJS[34], a popular JavaScript library for 3D geospatial visualization, VC Map combines the high-performance 3D rendering of CesiumJS with additional features tailored for complex data visualizations in urban planning, infrastructure management, and environmental monitoring. VC Map's architecture allows it to incorporate various map data sources, supports diverse data formats like GeoJSON, KML, and 3D Tiles, and enables users to easily switch between 2D and 3D views, enhancing the flexibility and utility of its mapping capabilities.

[34] JavaScript library for 3D geospatial visualization; https://cesium.com/

Figure 46: 3D Map

4.9.4.2 Key Features and Functionality

Interactive Visualization & Navigation

Multi-Dimensional Views: Seamlessly switch between 2D maps, 3D models, and oblique aerial images to gain different perspectives of urban areas.
Pedestrian Viewpoint: Examine scenes from a ground-level perspective to assess human-scale impacts.
Virtual Flyovers: Create dynamic camera flyovers to present city models to stakeholders effectively.

Analytical Tools

Drawing Tools: Add points, lines, polygons, circles, and text annotations in both 2D and 3D environments.
Measurement Functions: Calculate distances, areas, and heights directly within the map interface.
Height Profiles: Generate cross-sectional views along specified routes, incorporating all 3D objects present.
Visibility Analysis: Determine which objects are visible or obscured from specific viewpoints.
Shade Analysis: Assess shadow impacts on 3D objects at different times of the day, considering terrain models.
Clipping Planes: "Cut away" sections of buildings or terrain to reveal interior structures or underground features.

Data Exploration & Sharing

Search Functionality: Quickly locate addresses, street names, or cities within the map.
Split-Screen Comparison: Compare different datasets (e.g., meshes, 3D buildings, raster data) side-by-side using a swipe tool.
Transparent Surface Mode: Render specific areas transparent to visualize subsurface elements like pipelines.
View Sharing: Generate PDFs or share interactive scenes via direct web links for collaborative purposes.

Integration & Customization

Planning Tool Integration: Incorporate architectural models in various formats directly into 3D city models, enabling drafting and extrusion of buildings with floor-level detail.
Scene Export: Export scene into various formats, such as CityGML, CityJSON, 3D Shape, DWG, 3DS and more.
Open-Source Framework: Built upon open GIS and web technologies like OpenLayers and Cesium, allowing for customization and extension through a robust API.

4.9.5 Planner

4.9.5.1 Description and Purpose

The **VC Planner** empowers urban planners and city developers with a comprehensive suite of tools to create, analyse, and manage complex urban projects. Start by structuring project goals, phases, and milestones in a high-level overview, then leverage detailed 3D city modelling to visualize realistic urban environments. Integrate diverse spatial datasets, including GIS layers and BIM models, for a unified view that enhances data management and informed decision-making. The VC Planner supports the import of a wide range of data formats, including DWG, DXF, Shape, IFC, 3DS, OBJ, DAE, KML, KMZ, and GeoTIFF. Additionally, it integrates object libraries to enhance planning with features such as 3D tree libraries, 3D city furniture, 3D houses, and especially for URBREATH - the NBS catalogue.

VC Planner serves as a versatile platform for urban planning, enabling efficient design, analysis, and collaboration within 3D city models. Its comprehensive feature set supports various applications, from architectural competitions to collaborative urban development projects.

4.9.5.2 Key Features and Functionality

Rapid 3D Modelling: Transform 2D drafts into detailed 3D city models swiftly, facilitating quick visualization of urban planning concepts.

Intuitive Drawing Tools: Utilize tools to insert points, lines, polygons, circles, bounding boxes, and text labels, aiding in precise planning and annotation.

Dynamic Viewpoints and Camera Flyovers: Create custom viewpoints and camera flyovers with ease, enhancing presentations and stakeholder engagement.

Object Extrusion: Extrude drafted surfaces into 3D objects directly within the scene, streamlining the modelling process.

Import Capabilities: Easily import urban planning drafts, individual buildings, and street furniture using drag-and-drop functionality.

Display Management: Toggle the visibility of buildings and other elements with a simple click, allowing for focused analysis.

Customization Options: Design and fine-tune textures, line styles, point styles, and font formats to match specific project requirements.

Data Format Support: Supports various formats including shapefiles, glTF/glb, COLLADA, GeoJSON, KML/KMZ, and georeferenced raster plans. With the FME extension, additional formats like Autodesk 3DS, DXF/DWG, FBX, IFC, and OBJ are also supported.

Collaboration and Sharing

Project and Permission Management: Create multiple projects with distinct usage rights, facilitating collaborative work among different teams.

Web-Based Sharing: Share plans and 3D scenes via web links or publish them directly through a web browser, ensuring easy access for stakeholders.

Integration with VC Map: Manage planning objects within VC Map, allowing for seamless integration and visualization of planning scenarios.

4.10 E-Participation

Component diagram of the tools contained, with communication between them and possibly with other areas

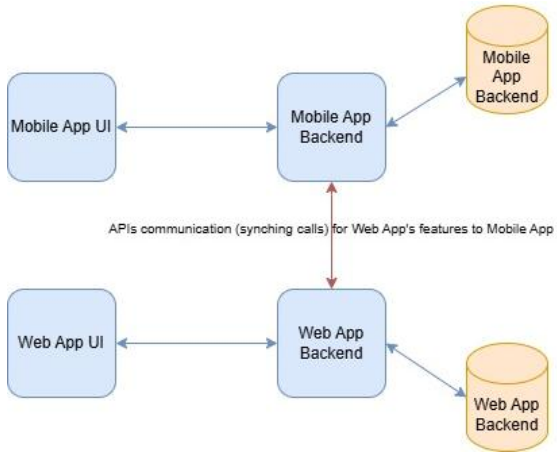
The e-participation solutions designed and implemented for URBREATH aim to support the "Mental Setup Change", a core element of URBREATH's vision, by fostering an open space of co-creation, co-designing, and co-decision making among diverse stakeholders of the city. URBREATH project recognizes that collaborative decision-making is critical to addressing the growing social, environmental, and infrastructural challenges facing European cities. E-participation concept has emerged as a powerful approach to engage diverse stakeholders in shaping sustainable interventions. The URBREATH digital e-participation solutions are designed and implemented to facilitate open discussions, idea exchange, and other functionalities that intend to bring together and bridge communication gaps among citizens, experts, academics, business owners, competent authorities, etc., into a collaborative environment.

These solutions are the two e-participation tools, the Web Application and the Mobile Application, both designed to facilitate the different needs of an open and co-creative environment for the diverse

stakeholders involved in everyday urban life to address collaboratively the challenges the urban areas face in the modern era. The two tools are briefly described in the following sections, and more details are documented in D4.4, "E-Participation Tools-V1", as well as their foreseen integration capabilities aiming to result in a comprehensive URBREATH digital e-participation suite.

The figure below visualizes the high-level architecture diagram of the two e-participation digital components and their communication.

Figure 47: E-Participation tools - Main components diagram



4.10.1 Web App

4.10.1.1 Description and Purpose

E-participation Web App offers a user-friendly digital environment supporting online interaction between different stakeholders, towards the implementation of co-creation activities and awareness creation concerning problems cities are facing (e.g. due to climate changes) and possible solutions to mitigate/solve them, helping cities in identifying urban interventions (e.g. NBSs) commonly agreed and that serve to address citizens' needs also.

The E-participation Web App is based on **Decidim**, which offers the possibility to setup different thematic participatory process empowered by components enabling different kinds of participation and contributions from the users, (e.g. publication of blogposts, comments, votes, feedback, debates, idea proposal and idea evaluation, participatory budgeting, meetings organisation, etc.). Those capabilities, offer a wide range of possibilities for engaging stakeholders into cocreation and participatory process. Further information concerning the capabilities and functionalities of Decidim are reported in "D4.4 – URBREATH Participatory tools - V1"

4.10.1.2 Key Features and Functionality

Decidim is an open-source tool whose aim is to foster participatory democracy by allowing citizens and in general stakeholders to interact directly public bodies and organizations and involve them into decision-making processes. Decidim is conceived to offer a transparent, accessible digital

environment where people can discuss, comment, propose, vote ideas, proposals, etc. and agree on shared policies and solutions for the community.

Decidim is based on modular functionalities that can be used to build a participatory process, such as for drafting a new urban plan.

A participatory process can include different phases; each phase can employ a specific functionality. For instance, during the proposal phase, users can leverage functionalities to write and submit their own ideas, which are then visible to the community, allowing other users to comment, suggest improvements, etc.

The debate functionality allows users to have conversations on different topics of interest, such as specific proposals. Furthermore, users are allowed also to vote proposals and allocate a virtual budget to different ones.

Another important functionality provides the possibility to organise meetings both in-person or online and publish the calendar of the different meetings that are organised on Decidim. This functionality allows also to access materials like agendas or minutes of a meeting, that can be published so everyone can be informed.

Finally, Decidim offers the possibility to organise assemblies. An assembly is a thematic space for ongoing collaboration (e.g. neighbourhood council, a committee, etc.), suitable for keeping long-term initiatives with transparency and coordination among their members, by leveraging above mentioned functionalities.

Detailed information functionalities Decidim offers can be found on its official documentation[35].

In addition to the functionalities already provided by Decidim (briefly summarised above), the URBREATH project included new ones, according to the plan reported in "D4.4 – URBREATH Participatory tools - V1". In particular, the current version of the E-participation Web App offers the following new capabilities:

- Integration of the Dataset Catalogue
- Integration of 3D Map (and NBS Storytelling)
- Integration of whiteboards
- Integration with the Security and User Management functionalities (i.e. Keycloak, section 4.11)

Integration of the Dataset Catalogue

The integration with the Dataset Catalogue allows the user to access the datasets collected and generated by the URBREATH Toolbox within the E-participation Web App. Through a dedicate

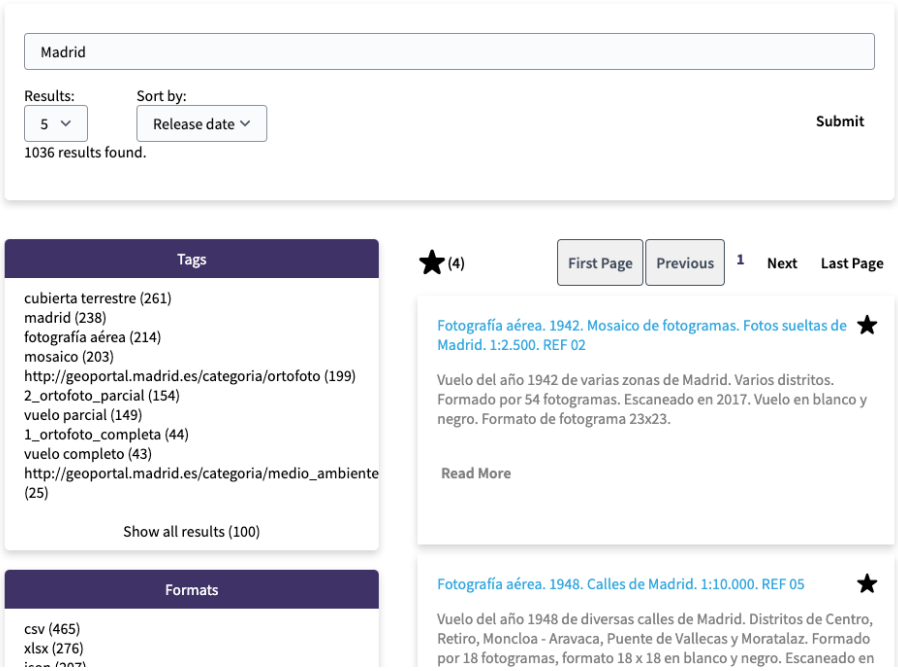
[35] https://docs.decidim.org/en/develop/index.html

section, the user can search among available datasets (by submitting keywords) and explore obtained results (i.e. datasets matching the submitted keywords).

Figure 48: E-Participation tools Web App - Integration of the Dataset Catalogue - Search

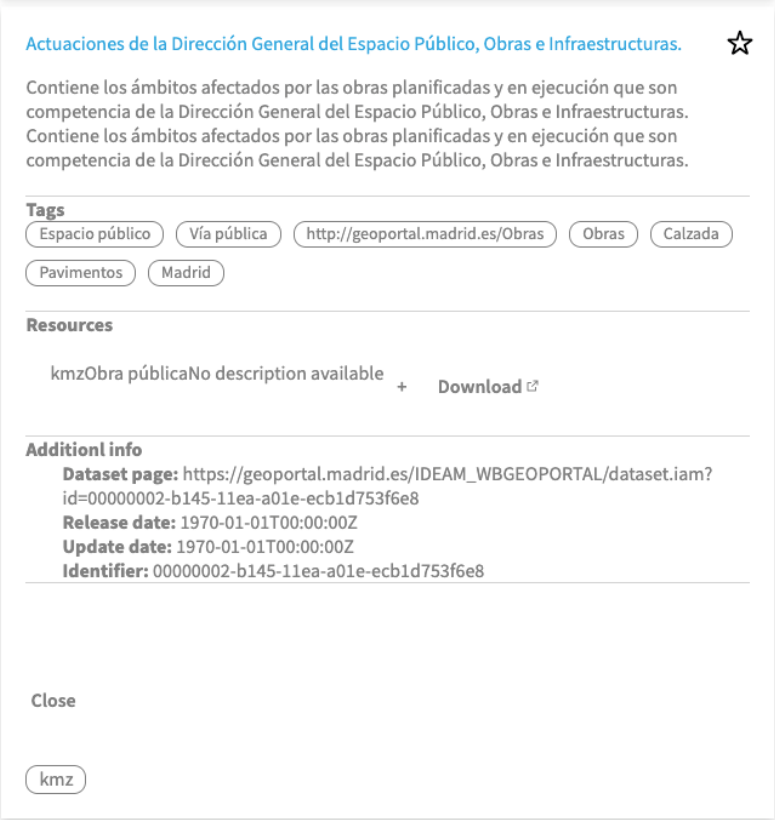
Results are provided in a list, and for each result the title and a preview of the description are reported.

Figure 49: E-Participation tools Web App - Integration of the Dataset Catalogue – List of results



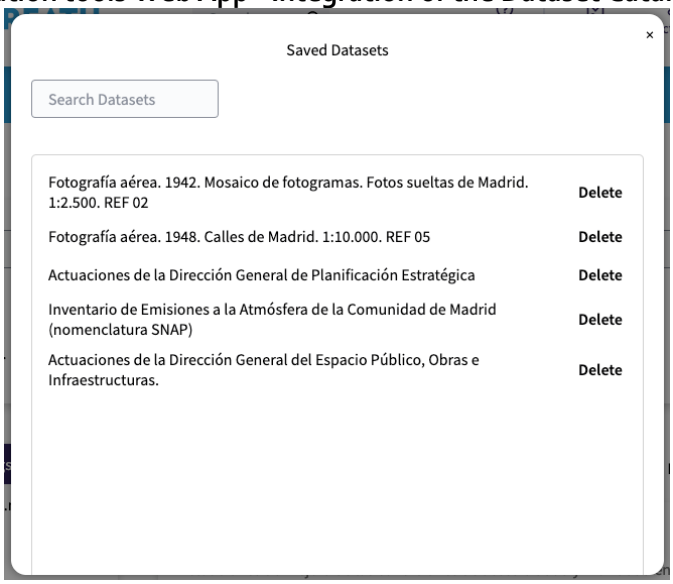
By clicking on the "Read More" button of a result, its details are displayed. Details include the title, the full description, the tags, available resources (i.e. concrete files) with the possibility to down them, the original web page of the result, the release and update dates of the result, and its unique identifier.

Figure 50: E-Participation tools Web App - Integration of the Dataset Catalogue – Details of results



Each result shows a star in the upper right corner (only when the user is logged in). By clicking on that star, the user can add the specific result among the favourite and access it later.

Figure 51: E-Participation tools Web App - Integration of the Dataset Catalogue – Starred result



The list of saved datasets can be also accessed from a test editor (e.g. when the user writes a comment), to easily include a reference to a specific dataset. In this case, the list reports the button "Add" for each saved dataset. Bu clicking on this button, a reference to the dataset is added in the text editor. Once the comment is submitted, other users can benefit from the linked dataset and access its content, supporting an evidence based participatory process.

Figure 52: E-Participation tools Web App - Integration of the Dataset Catalogue – Add reference to starred results (1 of 4)

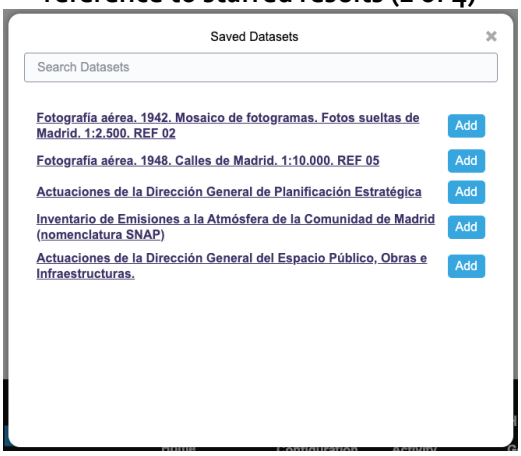
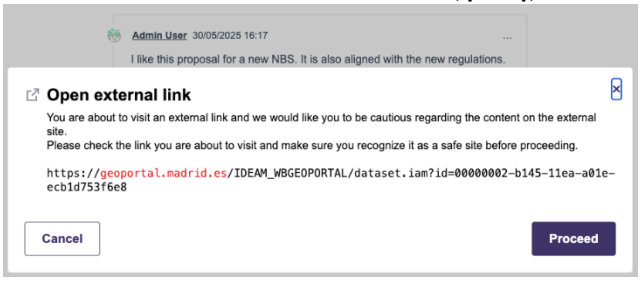


Figure 53: E-Participation tools Web App - Integration of the Dataset Catalogue – Add reference to starred results (2 of 4)

Figure 54: E-Participation tools Web App - Integration of the Dataset Catalogue – Add reference to starred results (3 of 4)

Figure 55: E-Participation tools Web App - Integration of the Dataset Catalogue – Add reference to starred results (4 of 4)



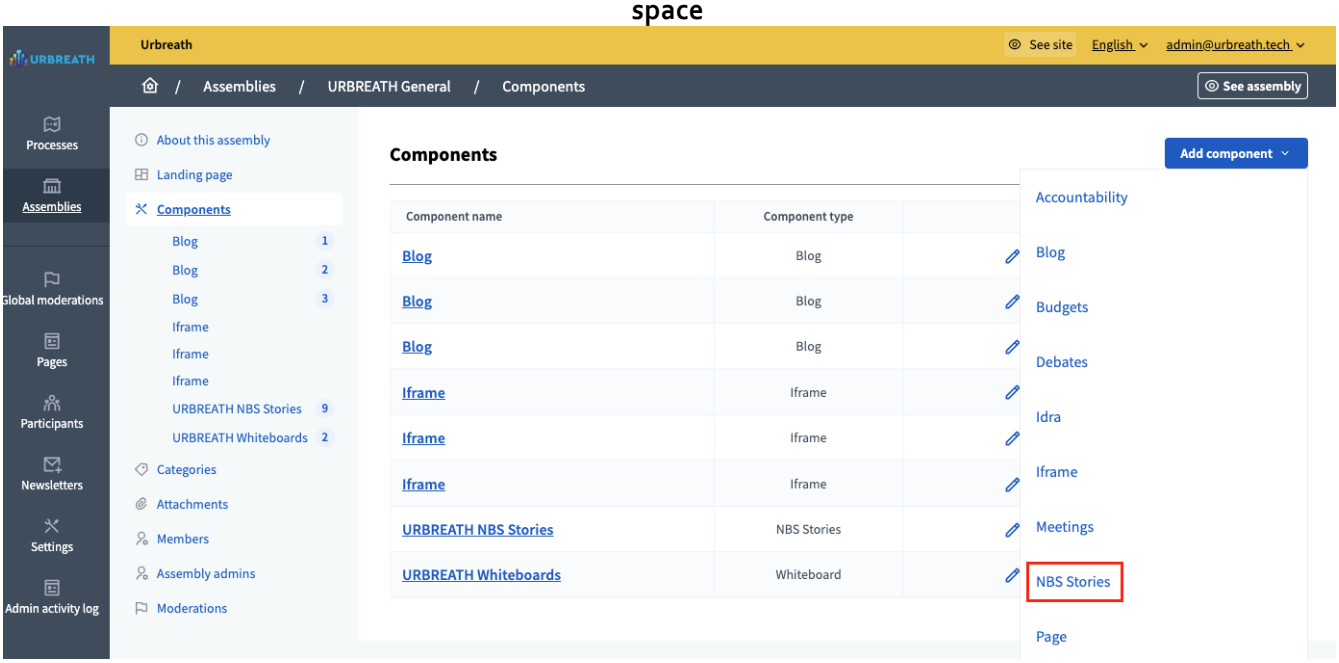
Integration of 3D Map (and NBS Storytelling)

This integration provides the users with the possibility to access 3D city models ad 2D/3D geospatial data (see section 4.11.4) directly within the E-participation Web App, enhancing evidence based participatory process. For this purpose, a specific plugin for Decidim has been implemented (named NBS Stories), which allows the administrator of the E-participation Web App to publish different 3D models.

The administrator first instantiates the NBS Stories component[36] into a participatory space (e.g. an assembly) and configure it (e.g. by setting the title, description, etc.)

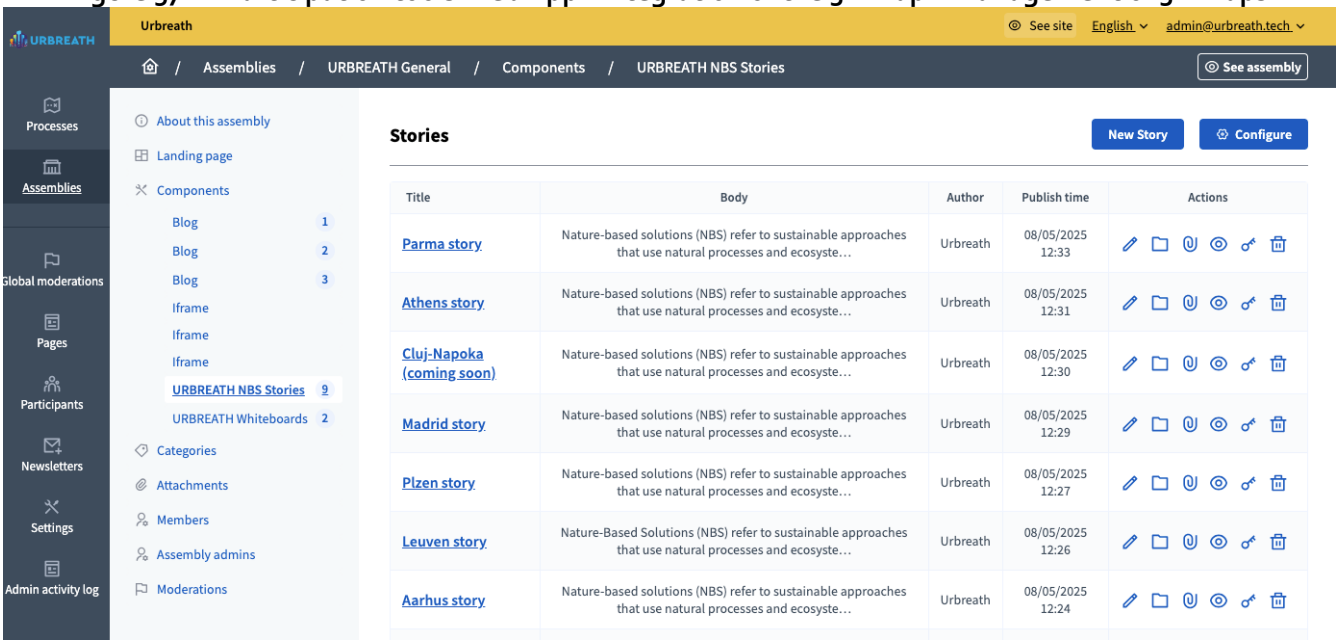
[36] In Decidim, the name "component" refers to participatory mechanisms; for more information, please refer to official Decidim documentation: https://docs.decidim.org/en/develop/features/components.html

Figure 56: E-Participation tools Web App - Integration of the 3D Map – Add 3D Map to a participatory space



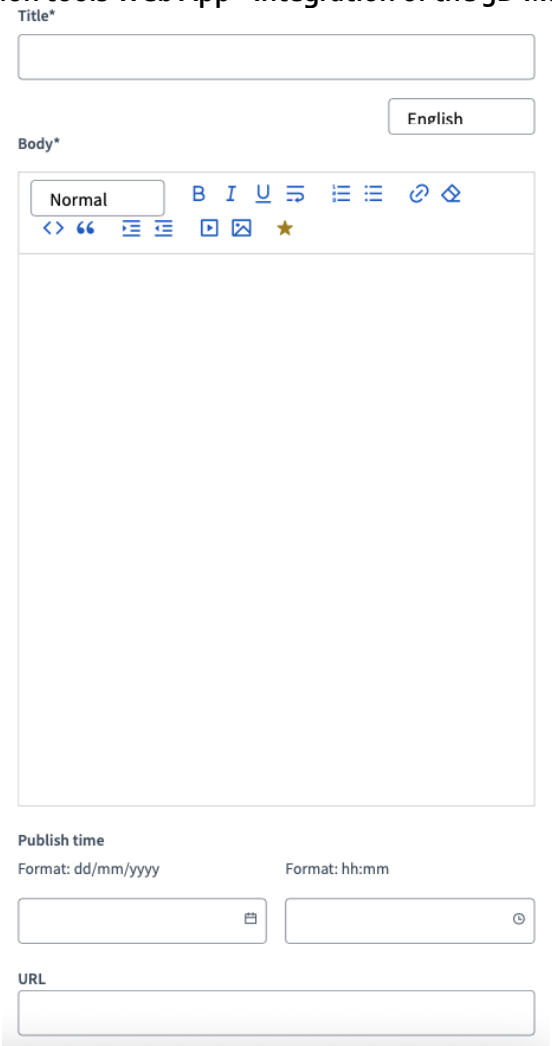
Once the component is instantiated and configured, the administrator can record NBS Stories (i.e. visualisation of 3D models and geospatial data, such as NBSs' representations). By clicking on the button "New Story", a specific form appears.

Figure 57: E-Participation tools Web App - Integration of the 3D Map – Management of 3D Maps



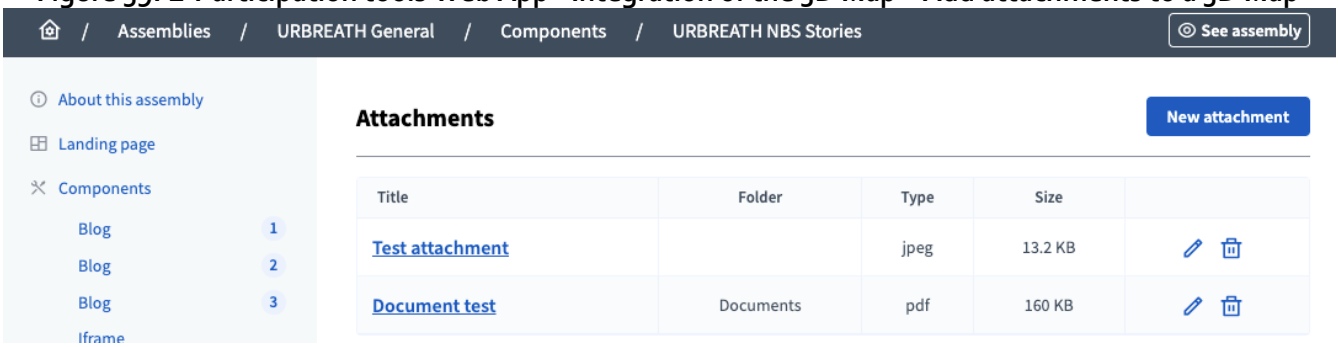
The form allows the administrator to insert information such as the title and the description (Body field of the form) of the NBS Story, as well as the URL to retrieve the 3D visualisation created through the 3D Map of the URBREATH Toolbox. It is important to underline that this functionality is integrated with the Dataset Catalogue also, since through the text editor of the Body field of the form the administrator can access his/her "starred" datasets and included references to them into the description (star button).

Figure 58: E-Participation tools Web App - Integration of the 3D Map – Register a 3D Map



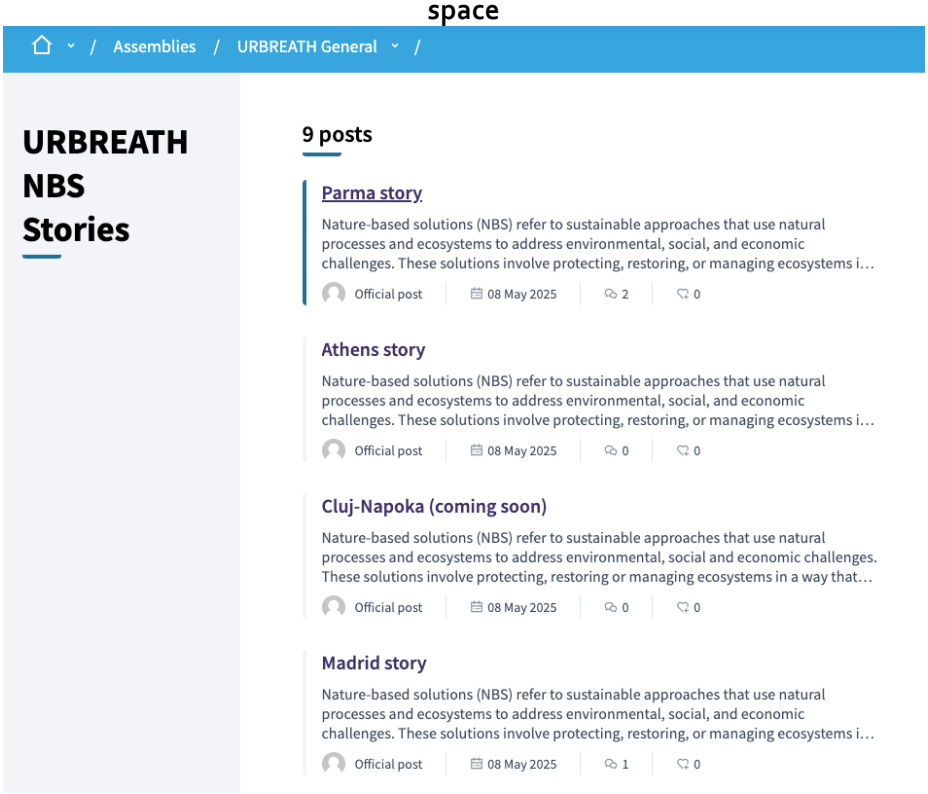
In addition, the administrator can also attached files such as images, documents, etc., that are made available to the users, that can be organised also in folders.

Figure 59: E-Participation tools Web App - Integration of the 3D Map – Add attachments to a 3D Map



Once the NBS Story is created, it is made available for the other user of the participatory space, where it is listed among other NBS Stories previously created. For each NBS Story, the list reports the title, the preview of the description, the author, the publication date, the number of comments, the number of "likes".

Figure 60: E-Participation tools Web App - Integration of the 3D Map – Explore 3D Maps of a participatory space



By selecting an NBS Story, the users can access its details, in order from top to the bottom of the page: the 3D design (the user can navigate and explore it), the description, the attached documents, and the comments (the user can add comments and reply to comments).

Figure 61: E-Participation tools Web App - Integration of the 3D Map – Visualise a 3D Map - 1 of 2

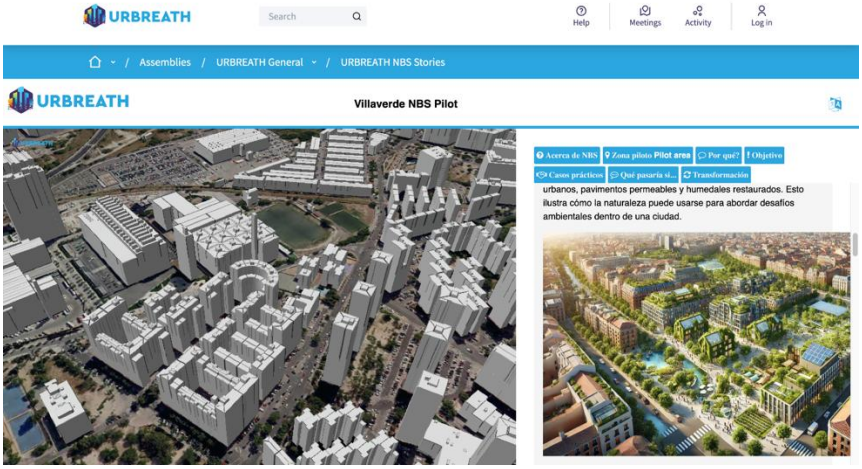


Figure 62: E-Participation tools Web App - Integration of the 3D Map – Visualise a 3D Map - 2 of 2

Figure 63: E-Participation tools Web App - Integration of the 3D Map – Visualise attachments of a 3D Map - 1 of 2

Figure 64: E-Participation tools Web App - Integration of the 3D Map – Visualise attachments of a 3D Map - 2 of 2

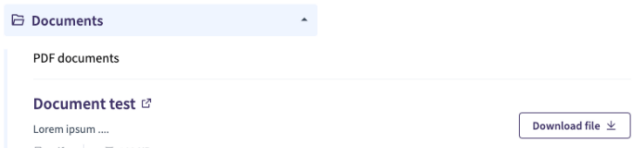
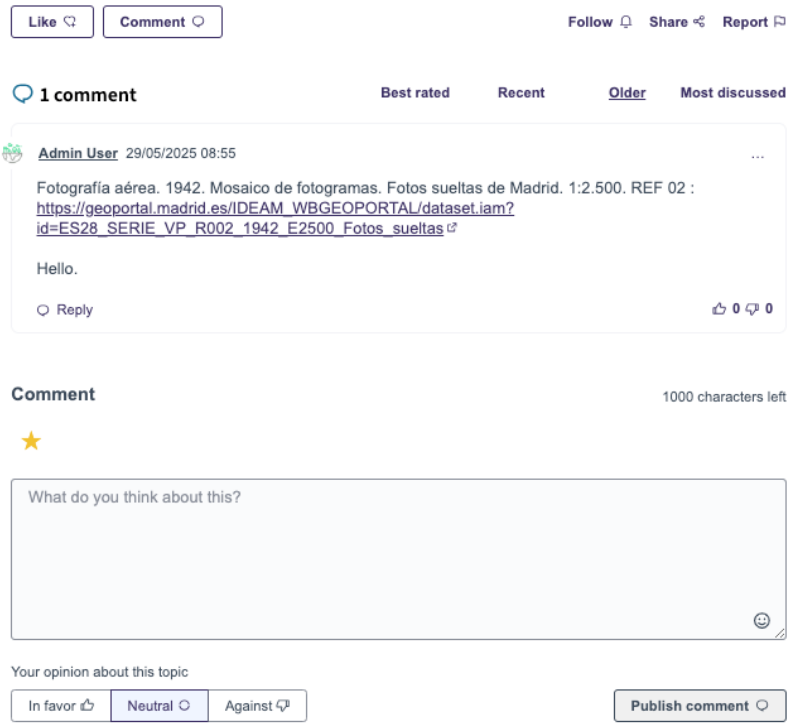


Figure 65: E-Participation tools Web App - Integration of the 3D Map – Comments management

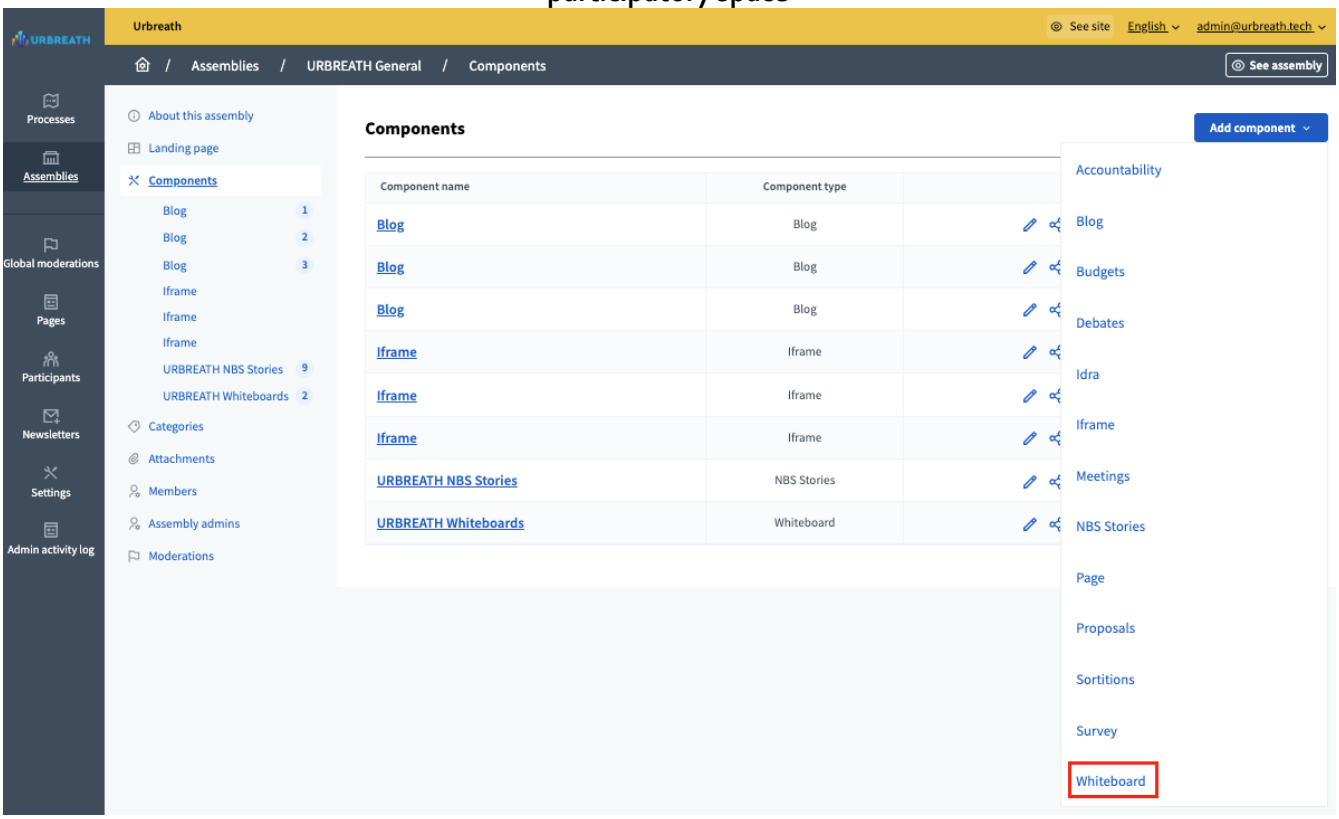


Integration of whiteboards

This integration provides the users with the possibility to access shared whiteboards directly within the E-participation Web App, offering relevant information and the possibility to have online collaboration, supporting interaction between users of a participatory space. A specific plugin for Decidim has been implemented (named Whiteboard[37]), which allows the administrator of the E-participation Web App to publish different whiteboards within a participatory space.

[37] According to its terminology, Decidim employes the term "component" for referring to plugin for participatory spaces.

Figure 66: E-Participation tools Web App - Integration of the whiteboards – Add whiteboards to a participatory space



The functionalities provided by this plugin (component for Decidim) are symmetrical to the ones related to the 3D Maps. The administrator can instantiate the Whiteboard components into a participatory space, register different whiteboard and configure them (i.e. set the tile, description, URL of the whiteboard, etc.), and attach relevant documents that can be organised into folders.

Figure 67: E-Participation tools Web App - Integration of the whiteboards – Management of whiteboards

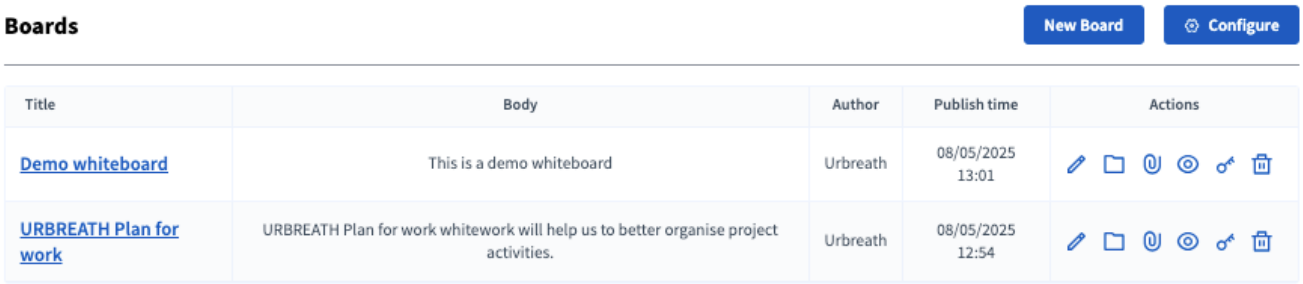
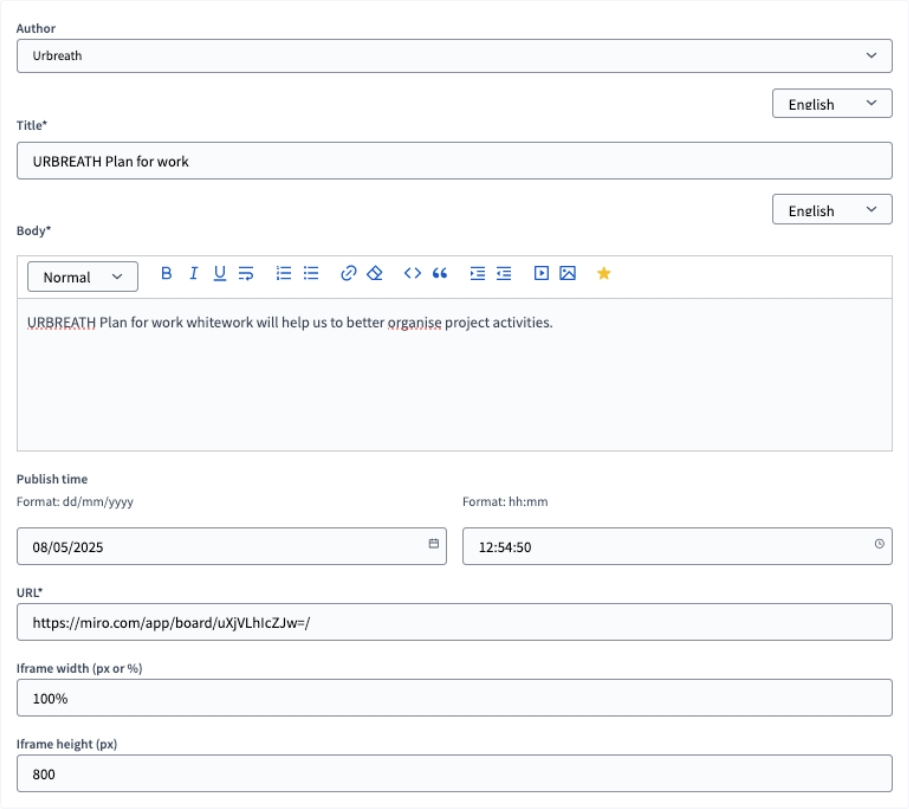


Figure 68: E-Participation tools Web App - Integration of the whiteboards – Register a whiteboards

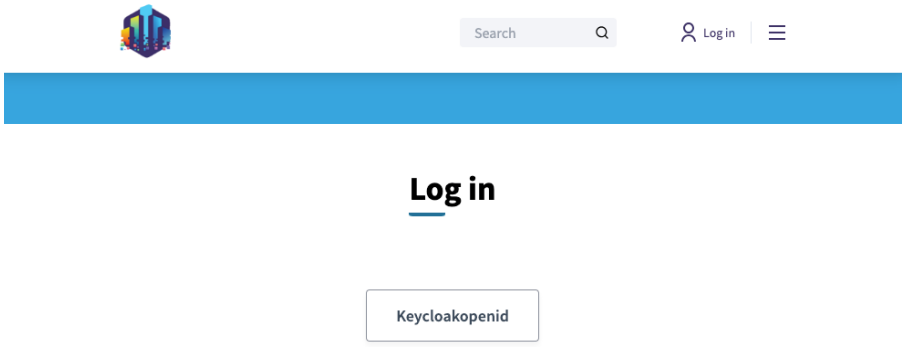


Integration with the Security and User Management functionalities

This integration allows the user to access the E-participation Web App by leveraging Single Sign On capabilities, by leveraging the capabilities offered by the Security and User Management macro area of the URBREATH Toolbox (section 4.11).

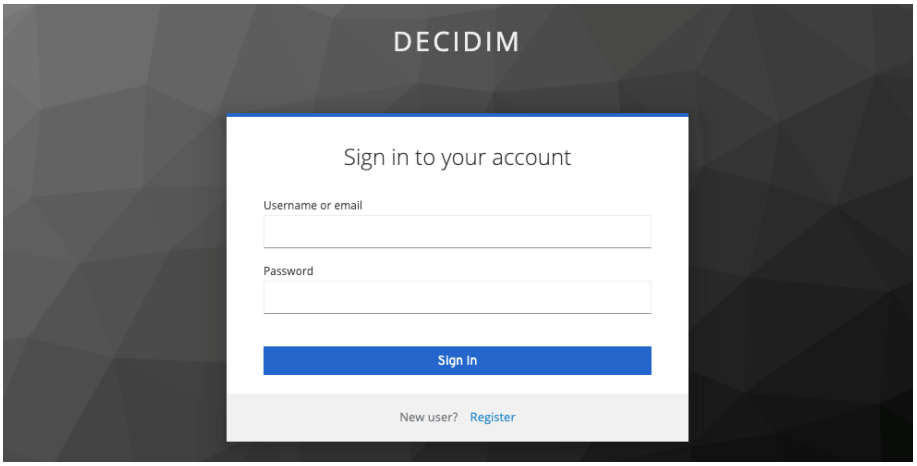
When the user requests to login into the E-participation Web App, the tool displays a dedicated page where the user can initiate the login process.

Figure 69: E-Participation tools Web App - Integration of the Security and User Management – Login page - 1 of 2



By clicking on the "Keycloakopenid" button, the user is redirected to the login page.

Figure 70: E-Participation tools Web App - Integration of the Security and User Management – Login page - 2 of 2



Once correctly logged in, the user is then redirected again to the E-participation Web App (i.e. Decidim).

4.10.1.3 Technical implementation

From a technical perspective, Decidim is built leveraging Ruby on Rails[38], a framework for the implementation of web applications by following the Model-View-Controller (MVC) architecture. Ruby on Rails allows to implement modular platforms, allowing developers to extend their functionality through additional components, which are distributed as "Ruby gems" (i.e. packages of reusable code that can be integrated as "plugins" into a Ruby on Rails application).

[38] https://guides.rubyonrails.org/getting_started.html

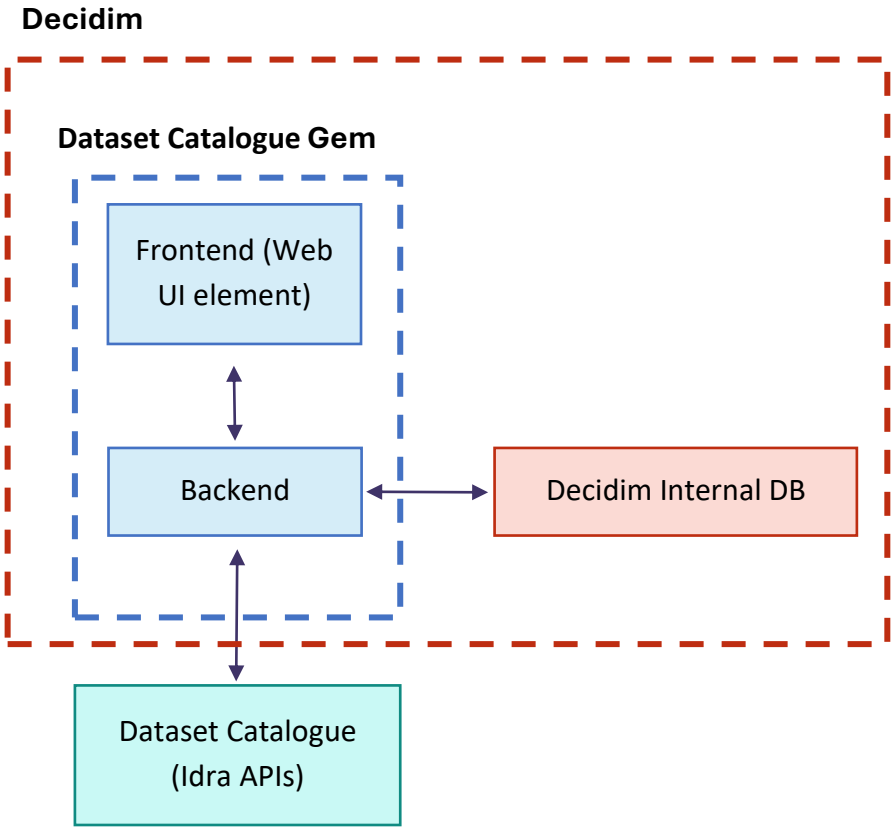
Decidim leverages this mechanism, and its functionalities are composed of different gems (e.g. proposals, meetings, participatory budgeting, etc.).

The integration of Decidim with Dataset Catalogue, the 3D Map, whiteboards, and the Security and User Management functionalities, are possible thanks dedicated gems.

Integration with Dataset Catalogue.

This gem includes both backend and frontend modules. The backend module manages the interactions with the APIs exposed by the Dataset Catalogue (see section 4.8.1), keeps track of users' starred datasets, and adds the access to the starred datasets from the text editor integrated into Decidim as well as from the comment section associated to published content (e.g. blogposts, proposals, etc.). To keeps track of users' starred datasets, this gem leverages the internal database of Decidim. Concerning the interactions with Idra's API, the backend module acts as an Idra's client application. The frontend module implements the UI elements that allow the users to interact with functionalities of the Dataset Catalogue (e.g. search and discover datasets, access their details, etc.), and with the specific functionalities for Decidim (i.e. management of starred datasets).

Figure 71: E-Participation tools Web App - Integration of the Dataset Catalogue – Schema



Integration with 3D Map and whiteboards

These gems provide the possibility to perform CRUD operations for managing content reporting 1) visualisations built on the 3D Map component of the URBREATH Toolbox and 2) whiteboards (e.g. employed for co-creation activities). Technically are both based on the "decidim-blogs[39]" gem and implemented as customisation of it.

Integration with Security and User Management

This gem provides Decidim with authentication capabilities offered by Security and User Management macro area of the logical architecture of the URBREATH Toolbox (see section 4.12). These capabilities are technically provided through the adoption of Keycloak.

This gem is based on an open-source project "decidim-module-keycloak[40]" w which has been customised for URBREATH to provide authenticated users (via Keycloak) with the possibility to create comments through Decidim APIs.

This gem is based on an open-source project "decidim-module-keycloak9" which has been customised for URBREATH to provide authenticated users (via Keycloak) with the possibility to create comments through Decidim APIs.

In particular, the gem now offers the possibility to verify the validity of the JWT[41] against the reference instance of Keycloak. Part of the integration with Keyclock is also the integration of an additional gem (JWT gem[42]) for decoding and verifying JWT tokens signed by Keycloak.

4.10.2 Mobile App

4.10.2.1 Description and Purpose

The e-participation mobile application is a tailor-made mobile application designed and created to provide the means for open communications among different stakeholders within a city or an area, promoting awareness and stakeholder engagement. This digital solution is aiming to support the communications and co-creation activities among citizens, experts, researchers, business owners, and other stakeholders, including the competent authorities involved in the city's decision-making processes.

39 https://github.com/decidim/decidim/tree/v0.29.1/decidim-blogs

[40] https://github.com/xdubx/decidim-module-keycloak

[41] JSON Web Token (https://www.rfc-editor.org/rfc/rfc7519); JWT is used to exchange authentication information between applications. For this purpose, JWTs are encrypted.

[42] https://github.com/jwt/ruby-jwt

In this sense, it provides a common space for all interesting parties to be involved in an open dialogue fostering co-design and co-decision-making for the interventions and strategic plans implemented in the city or suggested to be implemented, aiming to tackle major and minor issues that modern cities face (e.g., environmental issues due to climate change, neglected and abandoned areas that can be transformed into an area of value fostering NBS, etc.) in a collaborative manner.

By utilizing a common space for all stakeholders to share their ideas, needs, and perspectives and being part of decision-making, the city's strategic planning aspects, such as urban planning, become transparent, efficient, and more sustainable, as it takes into account the needs of all interested parties, enabling the plans to address the actual needs of the city as decided by all.

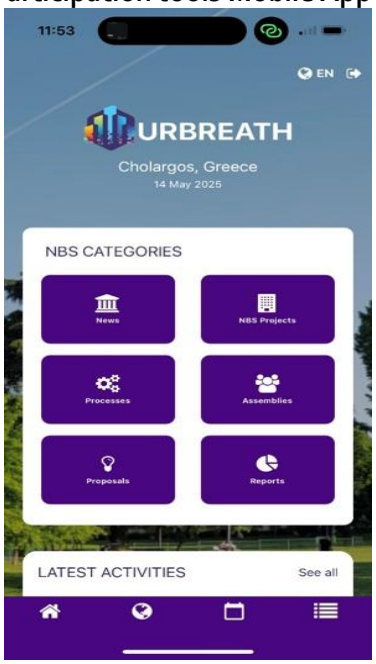
The e-participation mobile application is designed and implemented based on the friendly user experience logic and is fully customizable to meet and respond to generic or specific needs. It offers the ability for users to submit their proposals, report issues encountered within the city, and provide feedback on others' proposals, issues, or current projects. In addition, surveys can be leveraged by the authorities to gather input on specific issues and/or implementation plans. It is a native mobile application to promote efficiency in modern everyday life, whereas most people are on the move, thus allowing users to connect, interact, and engage through their mobile device everywhere.

Further information concerning the functionalities of the e-participation application is reported in "D4.4 – URBREATH Participatory tools - V1".

4.10.2.2 Key Features and Functionality

Aiming to facilitate an easily accessible and clear structure, the e-participation mobile app has incorporated different features, each of which aims to categorize a different perspective of co-creation and collaborative environment. The figure below shows the home page of the e-participation where end users can navigate to different features.

Figure 72: E-Participation tools Mobile App – Home page

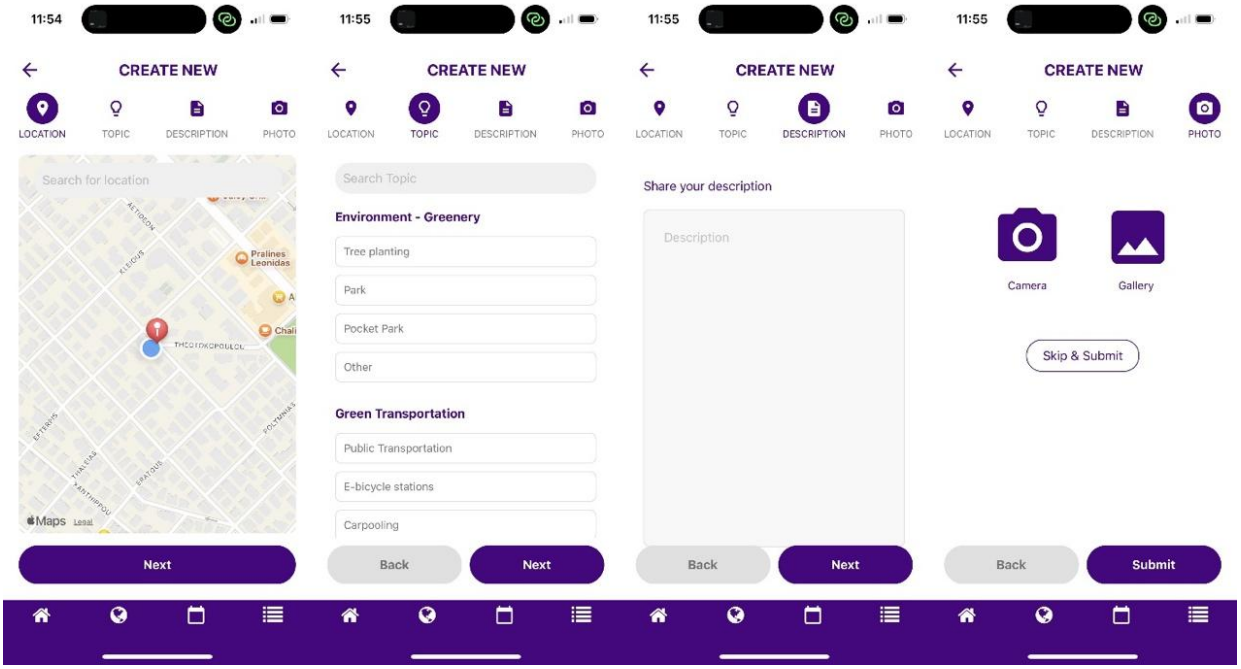


The current features are the result of the initial requirements as proposed by the pilot cities and requirements set by the Description of Action combined with a relevant feasibility assessment on how these requirements can be translated into technical components to facilitate the pilot's needs.

In particular, the current version of the e-participation mobile app includes these core built-in features and functionalities:

Ideas/Proposals on the Map: All registered users can submit their ideas/proposals for a specific location on the map. The process of submitting a proposal follows a step-by-step approach (Figure 73) where the users first indicate the preferable location on the map, then select a category and title, and then provide further details of their proposal. As a last step, there is an optional functionality where users can choose to incorporate a picture by attaching it from the device's files or using the device's camera to take a picture. Other users can support the proposal by adding "likes" or can provide a comment indicating alterations or enhancements, opening the discussion and collaboratively refining the proposal or co-creating a new one. Competent authorities can also participate in the discussions and collaborate, and they can also monitor the ideas/proposals, review the most preferable ones to decide if and how these proposals can be implemented in the city.

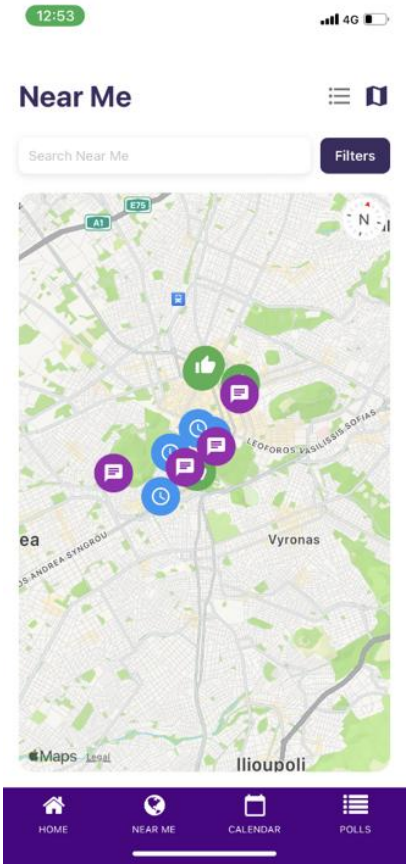
Figure 73: E-Participation tools Mobile App - Ideas/Proposals on the Map



- **Reports**: Registered users can submit a report detailing an issue of the city and its location on the map to inform others and competent authorities. This feature includes the category "invasive species" as an issue category. The process of reporting an issue follows the same logic as described for proposal submission, where the user selects a location, chooses a category or title, and provides further details. As a last step, optional functionality where users can choose to incorporate a picture is also available and incorporated within this feature. Other users can be informed and submit their comments, fostering an open dialogue among each other where competent authorities can also participate and collaboratively identify solutions or actions to be taken.
- **NBS Projects**: By accessing the NBS Projects, users can be informed about the interventions planned or already under implementation within their city. Users can review all details of the intervention, including location, timeframes, objectives, budgeting details, etc., that the competent authorities have made public alongside relevant attachments that may be included (e.g., pictures, plans, etc.). Support for the intervention can be indicated by the users by "likes" and a comment section where users can express their thoughts for their specific interventions and/or ideas is also included to maintain an open channel of discussion for the intervention and potential co-created adjustments.
- **Surveys**: Competent authorities can leverage this feature to gather input concerning specific policy-making plans or interventions such as NBS Scenarios, budget allocation aspects, and/or different aspects or issues concerning the city, including different alternative options for stakeholders to choose from. The application users can choose among the different options to state their preferences. This feature is also available for non-registered users as guest users, and it also provides a report with aggregated statistical results.

- **Near me**: In most features, the content regarding proposals, issues, NBS projects, etc. is accompanied by a location. By accessing the globe icon from the bottom ribbon on the home page, the relevant topics are translated as pins on the map. Users can select a "pin" to view the topic and can redirect to the specific space to view further details and engage with the content.

Figure 74: E-Participation tools Mobile App – Near me



In addition to its core features, the e-participation mobile application has incorporated features through its integration with the web app, which is based on Decidim as described in the previous sections. These features are:

- **Participatory Processes**: By connecting this feature from the web app, users can access the processes created and their components; thus, the users can keep up with the same content and activities both on the web and on mobile and interact from their mobile device and comment, sharing their perspectives and being part of the co-creation activities.
- **Assemblies:** Similarly, this integrated feature provides the same content and components incorporated, as designed in the web app, enabling users to be informed and be part of the relevant activities by interacting through their mobile devices.
- **Calendar:** This feature, created in the mobile app, facilitates quick access to the meetings created within the Web App's relevant component.

- **Common Keycloak service**: For simplifying the use of the e-participation tools, the same Keycloak service integrated in the Web app is also utilized in the mobile application so the users log in through the Web application.

As the project progresses, additional features may be developed according to pilot needs.

4.10.2.3 Technical implementation

As documented in D4.4, "e-participation tools-V1", the technical implementation of the e-participation mobile application started with its front-end development to move forward upon its foreseen integration with the Web application based on Decidim.

The app is built using React Native and Expo for cross-platform compatibility, with MobX employed for efficient state management. Communication between the mobile application and Decidim's API is established via secure HTTPS requests, utilizing a base URL that points directly to the Decidim API endpoint. Each request is carefully configured with the necessary headers and query parameters, and authentication is managed through JWT tokens to ensure secure, token-based access and real-time data synchronization.

The e-participation mobile application has then proceeded to the development of its own back end to support the app's core features and unique functionalities, employing a RESTful API architecture for robust data handling, secure storage and retrieval, and comprehensive user authentication and session management.

The technology stack can be listed as in Table 10.

Table 10: Mobile App – Technology stack

	Underlying technology
Front-end (UI)	React Native
Back-end	Node.js
Data layer	MongoDB
Bundling	EXPO GO
Authentication	Keycloak integration
Libraries	Axios Library

The technical implementation of the e-participation mobile application is an ongoing process as additional features may be developed and potential enhancements to already existing ones.

4.11 Security and User Management

This entire macro area of the URBREATH Toolbox logical architecture provides capabilities for user management, user authentication and authorisation.

Concerning user management, the administrator of the URBREATH Toolbox is allows to manage user identities, credentials, roles, and access policies. In particular, the administrator can create, update,

delete users (through a dedicated user interface) and allow self-registration of users (e.g. via email verification), reset users' passwords and configure credential policies (e.g. password strength, expiration time, etc.).

The functionalities provided by this macro area are based on Keycloak, an open-source identity and access manager (IAM) compliant with standard identity protocols such as OpenID Connect (OIDC), OAuth 2.0, and SAML 2.0.

The main features Keycloak offers are:

- **Single Sign-On (SSO)**: this feature allows users to log in once and access different applications avoiding the need to authenticate again.
- **Multi-Factor Authentication (MFA)**: this feature allows to improve security by supporting capabilities such as One Time Password (OTP).
- **Identity Brokering and Social Login**: this feature allows to integrate Keyclock with external identity providers (e.g. social media).
- **User Federation**: this feature allows to integrate Keyclock with existing user directories (e.g. LDAP) in use in an organisation.
- **Role-Based Access Control (RBAC)**: this feature allows to define and assign roles and permissions.
- **Session and Token Management**: this feature allows to monitor user sessions, configure token lifespans, and revoke access.

5 Integration environment

This section details the digital infrastructure supporting the integration of the provided digital services. It's based on a dedicated server and the software stack employed for the deployment and management of the URBREATH tools. Technical and functional aspects are outlined, encompassing hosting characteristics, the physical machine, operating system, server administration, and the strategy for release management, using Docker and Kubernetes.

The physical environment

The physical environment is built on a high-performance dedicated server. The project chose to opt for a dedicated server to have complete control over hardware, ensuring performance stability and enhanced application security.

The main characteristics of the employed server are summarised in the following.

- **Processor**: AMD EPYC™ 9454P (48 cores) for excellent multi-threading performance suitable for intensive and simultaneous processing.
- **RAM**: 320GB DDR5 ECC (256GB + 2x 32GB) providing ample, fast memory for concurrent demanding workloads. ECC[43] memory is vital for preventing data corruption and ensuring a stable, reliable environment.
- **Storage**: 2x 1.92 TB NVMe SSD Datacenter Edition (Gen 4) for fast-access data, significantly improving read/write speeds, and 2x 22 TB SATA Enterprise HDDs for large-volume, less-critical data storage. Two SSDs are configured in a RAID 1 array using `mdadm` with an EXT4 filesystem for maximum operating system protection. Two HDDs are configured in a mirroring setup using ZFS. ZFS offers continuous data integrity checks and error correction, providing enhanced security and reliability for large data volumes

This server configuration is selected for high performance in managing intensive workloads and mission-critical applications, such as development environments, complex databases, and high-availability containerized application hosting. ECC RAM enhances system reliability by minimizing hardware error risks and maximizing data integrity. The setup also supports scalability to accommodate evolving business needs without compromising performance or security. It is capable of handling multi-tenant environments, ensuring efficient resource sharing among applications or users while maintaining data isolation and protection

[43] Error correction code memory is a technology that allow to detect and correct corrupted data in memory

Software setup and server management

The adopted operating system is **Ubuntu Server 24.04**, that have been chosen for its reliability, security features, is capacity for managing intensive workloads, regular its updates and long-term support.

A dedicated software application is utilised to monitor the heath of the server: RedHat Cockpit. Cockpit is a powerful, user-friendly tool for server monitoring and management. Its interactive web interface provides real-time views of critical server parameters (CPU usage, RAM, storage, system logs) and facilitates maintenance tasks like package updates, service management, and disk and network monitoring, offering a visual approach for administrators.

Docker is the primary tool for release management of the URBREATH Tools, due to its ability to isolate applications in containers, ensuring a consistent and portable execution environment, which simplifies application deployment and scalability.

Key benefits of Docker include:

- **Isolation**: Applications run in isolated containers, preventing dependency and configuration interference and enhancing security.
- **Portability**: Containerized applications can be easily moved between different environments with consistency and reliability, ensuring identical code function across environments.
- **Efficiency**: Docker optimizes deployment times by reducing complex configurations.

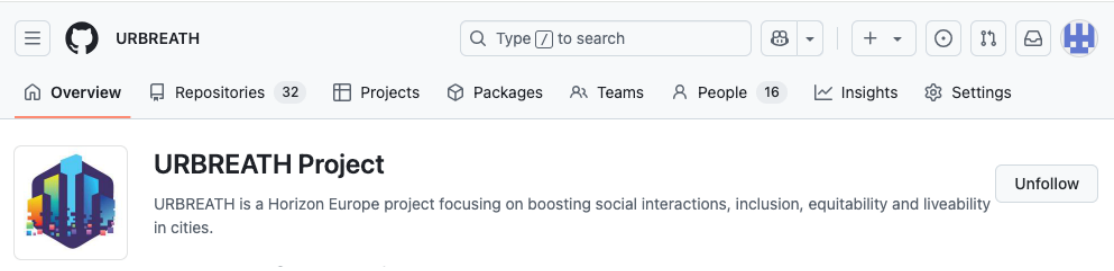
To further streamline Docker container management, **Portainer** is also employed. Portainer is a comprehensive management platform with an intuitive web interface, providing a user-friendly dashboard for monitoring, managing, and deploying containers, simplifying command-line Docker usage. Administrators can easily visualize container status, perform administrative tasks, and configure settings from a centralized interface, improving workflow efficiency.

In addition to Docker, also Minikube in employed for specific tools, such as the KPI manager, that need specific functionalities offered by Kubernetes.

6 Git Repository

Software components being part of the URBREATH Toolbox are documented in a collection of GitHub repositories belonging to the same "organisation" (URBREATH project[44]), a specific feature of GitHub that allows different contributors to collaborate.

Figure 75: URBREATH Git Repository



The GitHub repositories are used to collect and share

- documentation about software components
- source code (of open-source tools)
- Docker images or access to docker images of deployable tools

The strategy for sharing resources associated to each tool has been conceived flexible and adaptable to different cases, from complete open-source tool, up to tool accessible only as services (i.e. SaaS model)

For tool, the following cases have been identified, concerning resources to be shared.

- source code, Docker-compose file, Docker image (on GitHub Container Registry), and documentation, or…
- Docker-compose file and Docker image (on GitHub Container Registry), and documentation, or…
- Docker-compose file (make sure the Docker image is accessible) and documentation, or…
- complete running instance of the tool hosted in its own environment provided by the technical partner (SaaS model).

Each docker image can be:

- Uploaded to the public repository of URBREATH on GitHub[45], or…
- Hosted on a dedicated Docker Hub or GitHub Container Registry but must be accessible.

[44] https://github.com/URBREATH
[45] https://github.com/orgs/URBREATH/packages

7 Conclusions and Next Steps

This document summarises the initial version of the URBREATH Toolbox, reporting the main update about its logical architecture, the mapping between the personas identified in the deliverable "D2.5 URBREATH platform requirements", and the different digital services organised in the macro areas of the logical architecture.

Furthermore, this document also summarises the main technical characteristics of the integration environment employed for the implementation and testing of the components of the URBREATH Toolbox, and presents the URBREATH Git repository, where technical information and code of the open-source components of the URBREATH Toolbox are made available.

The URBREATH Toolbox will still evolve during the project, following the needs of the pilot cities and feedback collected from its adoption within them. Working in close contact with WP5, the technical work packages (WP3 and WP4) will have the chance to interact directly with the cities involved in the project and learn from the customisation of the URBREATH Toolbox to address local needs. This input will further feed and steer both the design of the toolbox (i.e. logical architecture and functionalities to be provided by the different tools that compose it) and the technical implementation of technological components.

Concerning the next steps, the technical implementation of the functionalities of the toolbox will focus on the following ones that are excepted on December 2025

In addition to the functionalities that are part of the initial version of the toolbox (section 9), the following ones are expected to be made available in December 2025. The implementation plan could be adapted according to changes in the needs and priorities of the pilots' cities.

City Data Visualisation

- Integration of real-time data visualisations into dashboards.
- Integration of the KPIs Manager into dashboards.
- Publication of reports from viewer analysis to NBS
- NBS creation and registration from LDT viewer
- Publication of report on the Dataset Catalogue from the Dashboard.
- Export charts as images (e.g. PNG format)
- Read/write comment on NBSs from the Digital Twin through the integration with e-Participation tool
- Read/write comment on NBSs from a NBS Registry through the integration with e-Participation tool
- 3D Map:
    - Visualise on map NBSs managed by the NBS Registry
    - Search and filter NBSs (on the map)
    - Access details of an NBS / Areas interested by NBSs

Data Analysis and Processing

- Management of simulations' parameters for the end user through the Unified UI[46]
- Pre-configured simulations' parameters for the end user (i.e. templates) through the Unified UI

E-Participation

- Publish KPIs on e-Participation Web App

[46] an overview of the analysis orchestration process is reported in section ABC

8 References

[1] EUROPEAN COMMISSION DIRECTORATE-GENERAL FOR COMMUNICATIONS NETWORKS, CONTENT ANDTECHNOLOGY Digital Excellence & Science Infrastructure Technologies for Smart Communities, “D05.02 Public report on the LDT Toolbox detailed specifications requirements”.

[2] Living-in.EU Technical Working Group, “MIMs Plus version 7 (2024),” 18 06 2024. [Online]. Available: https://living-in.eu/group/7/commitments/mims-plus-version-7-2024.

[3] Living-in.EU Technical Working Group, “Minimal Interoperability Mechanisms (MIMs Plus) - (Intermediate) Version 7.5,” 2025 02 04. [Online]. Available: https://living-in.eu/group/7/commitments/minimal-interoperability-mechanisms-mims-plus-intermediate-version-75.

[4] URBREATH Project, “D4.1 Local Digital Twin and KPIs catalogue for urban NBSs - V1”.

[5] URBREATH Project, “D2.5 URBREATH platform requirements”.

9 Annex A - URBREATH Toolbox 1st release functionalities

Data Discovery and Sharing

- Catalogue of Datasets
  - Publish and update datasets through a Web User Interface (UI) or APIs.
  - Search and discover published datasets through a Web UI or APIs.
  - 3D Map supports integration of datasets from IDRA, Piveau[47], and GeoNetwork.
- Catalogue of Tools & Models
  - Publish and update information about tools and models through a Web UI or APIs.
  - Search and discover published information about tools and models through a Web UI or APIs.
- Catalogue of Data Sources
  - Publish and update information about Data Sources through a Web UI or APIs.
  - Search and discover published information about Data Sources through a Web UI or APIs.
- Cross catalogues search & filter functionalities
  - Unique point of access to search and discovers relevant documents, etc, across multiple catalogues at the same time.
  - Filter search results to better identify relevant information (datasets, documents, models, tools, etc.); filters are automatically suggested, and the use can select which apply.
- Import 3D designs
  - Possibility to import existing 3D designs (e.g. from AutoCad) to be visualised on the Digital Twin.

City Data Visualisation

- Dashboard management
  - Users can create, edit, delete dashboards.
  - UI to build charts and dashboards quickly (no-code)
  - Web-based SQL Editor for querying connected data sources to build charts
  - Different built-in visualizations options to showcase data, (e.g. bar charts, geospatial visualizations, etc.)
  - Etc.
- Possibility to integrate visualisations of historical data into dashboards.
- Possibility to integrate KPIs into dashboards.

47 An open-source metadata catalogue; https://doc.piveau.io/general/introduction/

- Report management
  - Users can create, edit, delete dashboards.
  - Users can export a report in PDF and image (e.g. PNG) format for further use
- Extract data visualisation from the Dashboard management (e.g. to build presentations).
- 3D Map:
  - Draw /sketch in 2D/3D
  - Create fly-trough
  - Measurements in 2D/3D
  - Simulate shadows
  - Compare / swipe datasets
  - View data below terrain
  - Viewshed analysis (cone/360°)
  - Line-of-sight creation
  - Height – profile creation/measurement
  - Map out and plan NBSs
  - Share NBSs planning between users (to work together) [VC Planner]
  - Solar profitability calculator

City Monitoring

- Cities can integrate sensor data from Telraam and OpenAQ into the viewer.
  - Visualize measured data as diagram inside the viewer.
- KPI Management

Data Analysis and Processing

- Early versions of
  - 3-30-300 Analysis
  - Biotope Area Factor (small scale analysis)
  - Small-scale calc of BAF using the LDT viewer and planning tool.
  - Adaptive Rainfall-Infiltration Tracking
  - Urban Stress Analysis
  - Tree growing prediction
  - Shadow Modelling
  - 15 minutes cities (proximity index)
  - Visual Interpretable and Explainable AI

NBS Planning

- Catalogue of NBSs
  - Publish and update information about NBSs through a Web UI or APIs.
  - Search and discover published information about NBSs through a Web UI or APIs.

- Published NBSs can be associated to datasets managed by the "Catalogue of Datasets". providing fast access to relevant documents/information.

E-Participation

- Publish different multimedia content (e.g. text, images, documents, reports, dashboards, etc.).
- Publish 3D visualisation of Digital Twins.
- Implement participatory budgeting.
- Implement campaign to collect ideas.
- Users can comment published content.
- Manage events and publish their calendar.
- Users can access events' calendar from the mobile app.
- Implement awareness campaigns and support citizens' engagement by leveraging the possibility to publish content, participatory budgeting, ideas collection, allow comments and discussions, surveys, publication of events, etc. This is the result of different functionalities that together allows to implement awareness campaigns.
- Metrics about activities (e.g. number of participants, published posts, etc.).
- Notifications about relevant updates (e.g. received comments/answers, topics of interest, etc.)

Security

- Users and roles management
  - Create, update, delete users.
  - Create, update, delete roles.
  - Associate users and roles.

10 Annex B - URBREATH Toolbox logical architecture main updates

Wigh respect to D2.5, this document introduces some updates to the logical architecture of the URBREATH Toolbox, that are summarised in the following.

- GeoNetwork moved from the Data Management macro area to the Data Access and Sharing, since the GeoNetwork offer capabilities for the management of meta data of geospatial information.
- OGC Sensor Things API (Frost server) renamed in FROST Server.
- NBS Catalogue has been removed among the components of the Data Access and Sharing macro area of the URBREATH Toolbox logical architecture; its functionalities are now incapsulated into the NBS Registry and from a logical perspective it no more needed to report it as a separated tool. The NBS registry maintains interactions with the Dataset Catalogue.
- Air Quality and Pollution Levels Changes has been removed from the Data Analysis and Processing macro area; thanks to the availability of existing monitoring services, it has been decided to integrate those services with the URBREATH Toolbox offering the possibility to import and visualise data they already expose
- Snow Deposit Location Identification, Identification of best location for NBSs, Biodiversity Change analyses have been temporary removed from the Data Analysis and Processing macro area since further investigation with pilots are in progress with pilot cities.
- Two new analyses have been introduced in the Data Analysis and Processing macro area: 15 minutes city and Tree growing prediction.
- The "Data Spaces Interaction" has been renamed "Data Spaces Connector". The inner logical elements are not modified. Their concrete technical implementation is provided by TRUE Connector, an IDS certified open-source connector.
- The components Rule Manager, Nature Value Explorer, Cost benefit analyzer, Data Mashup Editor, and Satellite Image Connector, are not part of the current technical implementation of the URBREATH Toolbox; however, they are still part of the logical architecture, sine their technical implementation will be investigated during the course of the project.
- The macro area "Data Access and Sharing" has been renamed "Data Discovery and Sharing", to better reflect its purpose.

11 Annex C - URBREATH Toolbox position towards Minimal Interoperability Mechanisms and LDT Toolbox specifications

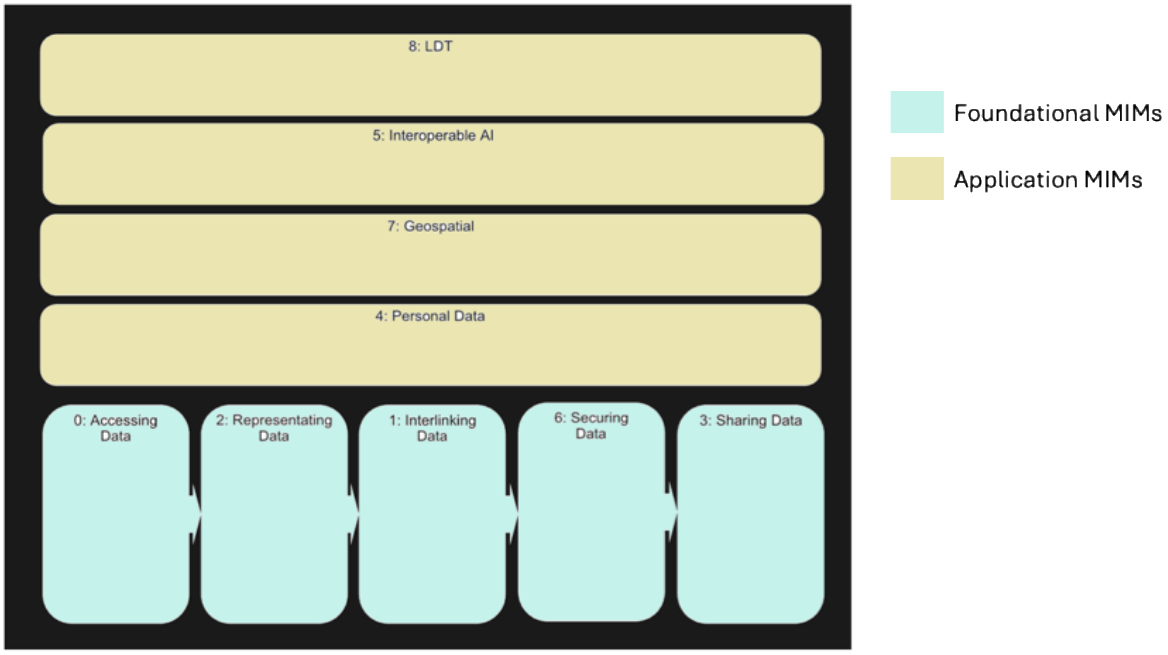
This section briefly summarises on how the URBREATH Toolbox moves towards the Minimal Interoperability Mechanisms (MIMs Plus) promoted by Living-in.EU[48] initiative and the LDT Toolbox specifications issued by EU Commission [1].

MIM Plus

Concerning MIMs, this section reports on how the URBREATH Toolbox addresses two versions of the MIMs: version 7.0 and version 7.5.

- Version 7.0 [2], released in June 2024, is the most complete currently available. Table 11.
- Version 7.5 [3], released in April 2025, introduces a new MIMs Framework based on Foundational MIMs (focusing on the data journey) and Application MIMs (addressing challenges such as personal data or local digital twins). This version of the MIMs is considered as "Intermediate" and it is still not complete. Table 12.

Figure 76: MIMs Plus (version 7.5) Framework Overview[49]



[48] https://living-in.eu/

[49] Original image from [3] has been modified for the purposes of the this document.

Table 11: URBREATH Toolbox position with respect to MIMs Plus version 7.0

MIMs[50]	URBREATH Toolbox position
MIM1 – Context Information • To enable context information from different systems within or across organisations, such as cities or communities, originating from heterogeneous sources, to be brought together using a Web based API. • To enable comprehensive and integrated use, reuse and sharing of data as well as management of context information • To turn data into a strategic resource	The URBREATH Toolbox includes a Data Management Layer that offers capabilities to interconnect with heterogeneous and scattered software systems and manage data coming from them building and uniform access to those data. Through connectors and tailored ETL process the URBREATH Toolbox can harmonise and aggregate data, that is then made accessible through standard formats and Web based APIs (e.g. ETSI NGSI-LD and OGS SensorThings APIs)
MIM2 – Data Models To support cities and communities to use consistent and machine-understandable definitions of all the entities about which data is being captured in a data ecosystem, along with a consistent set of identifiers of individual instances of each entity, so that data about any entity can be combined with other data referring to that entity, and every instance of that entity, in the confidence that they refer to the same thing.	The URBREATH Toolbox relies on two main standards that offers specifications for data models; OGS SensorThings APIs specification offers "SensorThings Data Model[51]". On the other hand, ETSI NGSI-LD specifications define a "meta data model" (i.e. NGSI-LD Meta Model[52]). Alongside, other common format is considered, to feed data analysis tools that require compact formats, such as CSV.
MIM 3- Contract • To provide data along with relevant information about its content and quality and any terms and conditions for use. • To provide data processing services along with relevant information and terms and conditions for using the services. • To find and access the data and data processing services and other services they need and to be able to gain relevant insights into what those data streams/data processing services/data applications consist of and how valuable they can be.	The URBREATH Toolbox includes three catalogues: Dataset Catalogue, Models/Tools Catalogue, Data sources Catalogue. They offer capabilities for the management and search of relevant information associated to • datasets (e.g. coming from relevant systems connected to the toolbox and/or produced by the analysis that are part of the toolbox). • models and tools available within the toolbox (including the ones for data analysis). • data sources connected to the toolbox. Information managed by the three catalogues follows the DCAT-AP and are managed also on the Context Broker as NGSI-LD entities.

[50] For each MIM, a short description extracted from [1] is reported.

[51] https://ogc-iot.github.io/ogc-iot-api/datamodel.html

[52] https://cim.etsi.org/NGSI-LD/official/9-tabcontext-information-management-framework.html#tabngsi-ld-meta-model

	For the sake of clarity, it is important to report that MIM3 covers aspects also related to offers and order, revenue sharing, service level agreements, feedback and reputation, party management, customers, transparency and accountability, that move towards the establishment of "marketplaces", that are out of the scope of URBREATH. For this reason, the URBREATH Toolbox, focused mainly aspect related to the establishment of catalogues.
MIM 4 – Trust To enable individuals to be able to easily manage data about themselves so that it can enable outcomes they want, both for themselves and their community, while not compromising on privacy. To do this in a way that will make it easy to integrate with whatever credible personal data management systems (such as forthcoming EU-registered personal data intermediary services) the individual may wish to use.	Despite the general relevance of the topic, the URBREATH project does not focus on personal data management, and currently the toolbox does not include a tool for this purpose, neither this point emerged from investigations performed with the cities. However, the URBREATH project recognises the importance of this topic. The dialogue with the nine cities involved in the project is still in progress and will continue until the end of the project; if the need for IT solutions suited for allowing end users to manage personal data raise, the project will commit to identify one or tools that can be employed for this purpose.
MIM 5 – Transparency The objective of MIM5 is to identify a set of minimal, but sufficient, mechanisms to help cities and municipalities have the technical capabilities to verify that the algorithmic systems offered by providers to make decisions about the services provided to citizens are fair, trustworthy and transparent. • This includes the models used, the datasets on which the algorithms are trained, and the data that the models use to make decisions. • The current starting point for assessing the fairness, trustworthiness and transparency of those algorithms will be the AI procurement clauses originally developed by Amsterdam.	The URBREATH Toolbox offers an online technical documentation available in a dedicated public Git repository, for each tool that is part of the toolbox. In addition, a specific tool (i.e. VIE-AI, section 4.5.8), that, given a AI-based model, can offers insights on how various input affect model predictions.
MIM 6 – Security	One of the aims of the URBREATH Toolbox is to offer a series of IT tools that can be employed within existing cities' platforms according to

• When information is transferred, between parts of the data platform or externally, this is done securely. • Data processors know what requirements concerning security and interoperability to make of suppliers and systems when evaluating, procuring, developing, operating, and using solutions.	specific needs (e.g. by selecting and integrating specific ones) and/or by adopting the entire architectural stack, and related tools. In the first case, security aspects concerning internal data transfer should be addressed considering existing technical approaches and in compliance with possible local policies, regulations, technical prescriptions, etc. in force. In the second case, the URBREATH Toolbox can leverage the adoption of secure communication channels between its components (e.g. SSL); furthermore, the toolbox includes solutions for user and role management, that allows to define who can access specific resources. Concerning he possible transmission of data with external platform, the toolbox includes software solutions for the interconnection with Data Spaces (see section 4.2.5).
MIM 7 – Places To enable cities and communities to easily integrate data about spatial assets such as streetlights, buildings, and streets with spatio-temporal data from sensors, along with other data sources that can provide helpful context information to the geospatial data, and make the data interoperable within, and between cities and communities.	The URBREATH Toolbox employs tools dedicated to the management of geo-spatial information, GeoServer (section 4.1.4) and GeoNetwork (4.8.2). The first is able expose WFS compliant interfaces to access geospatial information, whereas the second to manage metadata associated to this information. In addition, the GeoCacher (i.e. Map Layers Manager, section 4.9.3) can generate map layers (in GeoJSON format) from NGSI-LD compliant entities managed by the Orion-LD Context Broker (section 4.3.2). GeoCacher can also export those layers into the GeoServer, making them available through its APIs. The adoption of both Orion-LD Context Broker and FROST Server (compliant with OGC SensorThings APIs) allows to link and integrate geospatial and spatio-temporal information.
MIM 8 – Indicators Cities and towns are complex systems, and no two cities or towns are identical in the scale or scope of their complexity. Still, there is increasingly clear value when cities benchmark some measurements against comparable peer	From the technical perspective, the URBREATH Toolbox includes the KPI Manager (section 4.6.1), a tool dedicated to the management of KPIs. The KPI Manager offer simplified functionalities to define:

cities, as well as learn from the success and failures of other cities. Innovation ecosystems are no exception. Ecosystem indicator management allows: • Development of consistent measures of the ability of different cities to provide a healthy and effective ecosystem that nourishes digital transformation and supports interoperability of data, systems, and services, • Governing of cities performance against these measures, • Benchmarking of results and practices among comparable peer cities, • Planning, deploying, and monitoring ecosystem improvement initiatives	• measures of interest by aggregating data from various sources employed in the KPI calculation. • custom KPIs by defining their formulas and associating with measures of interest. • set target of KPIs and monitor deviations. • Visual representations (charts and graphs).
MIM 9 – Analytics Aims to make complex data models interoperable, allowing more efficient analytics and impactful exchange of expertise, to allow cities to leverage each other's successes in data analytics.	The URBREATH Toolbox considers, among the other, the following ETSI NGSI-LD, OGC SensorThings API, OGC WFS, and CityGML [4] specifications to represent and access managed information, that allow to manage complex and interoperable data models. In addition, the current version of the toolbox includes the "Workflows Management and Execution" (i.e. Apache Airflow), which is able to manage complex workflow as python scripts, giving the change to replicate those workflows across cities.
MIM 10 – Resources Interoperable capabilities related to management and assessment of scarcity and resources related to people, nature, and investment.	As foresaw in [5], the URBREATH Toolbox will include tools to support the estimation of the effect of NBS interventions from the perspective of the planning, in addition to the NBS Registry. These tools are the Nature Value Explorer (for mapping the socio-economic importance of nature), and the Cost Benefit Analyzer (to evaluate cost and benefit distribution of possible interventions).

Table 12: URBREATH Toolbox position with respect to MIMs Plus version 7.5

MIMs[53]	URBREATH Toolbox position
Foundational MIMs	
MIM 0 – Accessing Data Ensuring that data is accessible across diverse systems of an organisation via interoperable APIs. This MIM helps cities and communities to overcome the challenges of accessing data that currently reside in silos and are often fragmented across different systems and locked away by making recommendations of how data should be made accessible so it can be easier accessed and reused in an interoperable way.	The URBREATH Toolbox includes solutions for data management based on well-known open-source project that offer documentation and API specifications. Among them, it is important to mention the FROST Server (that follows OGC SensorThings APIs) and the Orion-LD Context Broker (that follows NGSI-LD Specifications). Furthermore, the toolbox offers technical solutions to build catalogues of datasets, software models/tools, data sources, and Nature Based Solutions (that are the focus of the project).
MIM 1 – Interlinking Data Ensuring that data sources can be interrelated to each other based on their contextual relationships in an interoperable way to enable improved interpretation and exploitation of the data. This MIM helps cities and communities overcome the challenges of disconnected data sources that reside in different contexts of diverse systems. It provides recommendations on how diverse data sources can be contextualised to a city and community's operational environment, consistently combined, interrelated, and interpreted so meaningful insights can be derived.	As reported Table 11 concerning "MIM1 – Context Information" and "MIM2 – Data models", the URBREATH Toolbox offers tools for Data Management Layer able to interconnect with heterogeneous and scattered software systems, establishing a uniform access to those data. Connectors and tailored ETL process can be employed to harmonise, aggregate, and interlink data, leveraging standard formats and Web based APIs such as ETSI NGSI-LD and OGS SensorThings APIs.
MIM 2 – Representing Data Ensuring data can be more effectively used within an organisation by representing it using standards-based interoperable data formats. This MIMs allows cities and communities to overcome the challenges of working with data that are represented in different ways by providing recommendations for the use of consistent and machine-readable representations of data so that data from	Please refer to URBREATH Toolbox position reported in Table 11 concerning "MIM2 – Data Models".

[53] For each MIM, a short description extracted from [1] is reported.

various sources can be efficiently used with confidence across the organisation and shared with collaborators as part of a local data ecosystem.	
MIM 3 – Exchanging Data Ensuring that data can be effectively discovered by data users across an organisation and stakeholders in a wider data ecosystem, exchanged in interoperable ways to enable data reuse. This MIM helps cities and communities set up a local data ecosystem that is globally interoperable. It helps overcome the challenges of how data can be "pooled together" across diverse systems of different organisations into a common data space that enables data providers and users to exchange data in a trusted way with confidence and derive value from it.	Please refer to URBREATH Toolbox position reported in Table 11 concerning "MIM3 – Contract" and "MIM6 – Security".
MIM 6 – Securing Data Ensuring that data is adequately secured and protected within a data ecosystem (in storage or during transit) in an interoperable way. MIM 6 focuses on addressing interoperability for secure data transfer. When information is transferred, between parts of the data plat form or externally, this is done securely. Data processors know what requirements concerning security and interoperability to make of suppliers and systems when evaluating, procuring, developing, operating, and using solutions.	Please refer to URBREATH Toolbox position reported in Table 11 concerning "MIM6 – Security".
Application MIMs	
MIM 4 – Personal Data To enable individuals to be able to easily manage data about themselves so that it can enable outcomes they want, both for themselves and their community, while not compromising on privacy. • To enable individuals to be able to easily manage data about themselves so that it can enable outcomes they want, both for	Please refer to URBREATH Toolbox position reported in Table 11 concerning "MIM4 – Trust".

themselves and their community, while not compromising on privacy. • To do this in a way that will make it easy to integrate with whatever credible personal data management systems (such as forthcoming EU-registered personal data intermediary services) the individual may wish to use.	
MIM 5 – Interoperable AI The definition of this MIM is not yet available.	Not available since a definition of this MIM is not provided.
MIM 7 – Geolocation MIM7 aims to provide Minimal Interoperability Mechanisms related to geospatial data, to tackle the challenge faced by cities and communities of being able to integrate and transfer data between internal and external IT systems. It also considers the fact that spatial assets need to be accessed as linked data by many IT- and IoT-systems, and over a long period of time, and thus the vital role of the use of persistent identifiers. • To enable cities and communities to easily integrate data about spatial assets such as streetlights, buildings, and streets with spatio-temporal data from sensors, along with other data sources that can provide helpful context information to the geospatial data, and make the data interoperable within, and between cities and communities. • This integration should be made possible across technologies and vendors.	Please refer to URBREATH Toolbox position reported in Table 11 concerning "MIM7 – Places".
MIM 8 – LDT The definition of this MIM is not yet available.	Not available since a definition of this MIM is not provided.

LDT Toolbox specifications

The LDT Toolbox specifications aim to accelerate the digital transformation of European cities; the specifications offer guidance for both the implementation of the LDT Toolbox itself but enable also the cities to build and deploy Local Digital Twins.

For this purpose, the specifications include a guiding reference architecture to support cities in building their LDT, and underpin the principles of openness and interoperability, pushing towards the

possibility that cities of all sizes can adopt and adapt enabling technologies for the implementation of LDT, including standards and further specifications such as the Minimal Interoperability Mechanisms (MIMs).

The LDT Toolbox Specifications includes 18 building blocks that are depicted in Figure 77. Table 13 provides a short description of each building block and the related URBREATH Toolbox position.

Figure 77: LDT Toolbox specifications Reference Architecture

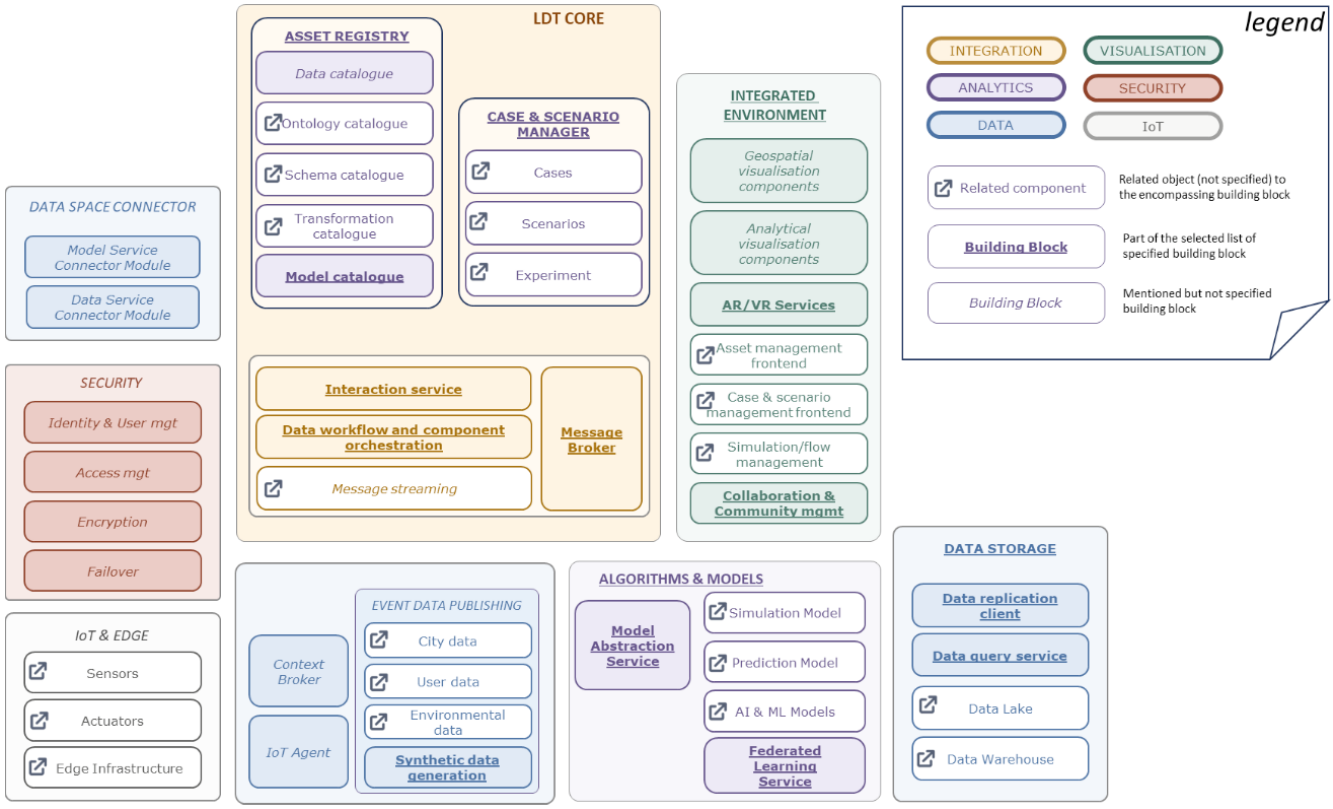


Table 13: URBREATH Toolbox position with respect to LDT Toolbox specifications

LDT Toolbox Building Block (BB)	URBREATH Toolbox position
BBo.1 Reference Architecture The reference architecture (Figure x) aims to create a joint vision and understanding of the concepts and components that compose the system.	The URBREATH Toolbox offers a logical architecture (section 2) which envisages macro areas and components that aligns (in part or as a whole) with the concepts BB.01 represents. Among them, as examples, • Datasets, Models/Tools, and Data sources catalogues that follow the Asset Registry (BB.16) and Model Catalogue (BB.13); • The URBREATH Toolbox Message Broker aligns with LDT Toolbox Message Broker

	(BB.11), since Apache Kafka is mentioned among the suggested tools. • The NBS Planning macro area and the NBS Registry follow the Case & Scenario Manager. • Etc.
BB.02 Case & Scenario Manager It allows to manage cases and scenarios. Cases are specific problems to be addressed (e.g. planning an urban intervention); scenarios collect and evaluate the possible solutions, data, simulation models, etc.	The NBS Planning macro area aims to offer functionalities to support decision makers, planners, etc. in investigating, designing, and planning NBSs. Among the tools of the first version of the URBREATH Toolbox, the NBS Registry aims to offer functionalities for tracking, verifying, and managing NBS related projects.
BB.03 Collaboration and Community Management System This BB aims at involving the public in decision-making, policy development, and service improvement, facilitating collaboration and creating community practices.	The e-Participation tools (i.e. the Web App and the Mobile Application; section 4.10), offer functionalities for enabling stakeholders' engagement and collaboration.
BB.04 Data Query Service This BB offers an interface to access, explore, and understand the data within the Digital Twin, providing an abstraction over underlying data storage systems, query services ensure efficient, optimisation access to data.	The Data Repositories Connectors of the URBREATH Toolbox offers an abstraction layer enabling uniform access to heterogeneous databases. A further alignment element is represented by Presto, which is the tool employed by the URBREATH Toolbox and it is among the tools suggested by LDT Toolbox Specifications.
BBo.5 Data Replication Client It offers capabilities for maintaining an accurate and up-to-date representation of a city in a local Digital Twin, by capturing changes from different asynchronous sources (e.g. output of models, IoT platforms, physical environment changes, etc.).	The URBREATH Toolbox includes a series of connectors, which aim is to establish connection point between the IT system of the city and the toolbox, enabling continuous data capturing also from asynchronous sources. In addition, the adoption of solutions such as Apache Kafka (Message Broker) Orion-LD (Context Broker) enables message stream management and publish / subscribe capabilities, towards data replication.
BBo.6 Data Workflow and Component Orchestration This BB offers functionalities to execute workflows made of multiple and interrelated steps in the right order, facilitating the data	The URBREATH Toolbox includes a component which is dedicated to data workflow and components orchestration, that is based on Apache Airflow. Apache Airflow is among the tools suggested by LDT Specification for the implementation of this BB.

processing for deriving valuable insights or actionable information.	
BB.07 Extended Reality This BB allows to implement immersive experiences for the users, enabling them to interact with virtual elements.	Currently, the URBREATH Toolbox does not include this kind of capability.
BB.08 Federated Learning Service It allows to train models across multiple decentralised edge devices or servers holding local data samples, without explicit exchange of the data.	Currently, the URBREATH Toolbox does not include this kind of capability.
BB.09 Integrated Environment This BB offers a unified and consistent user experience that enhances usability, efficiency, and accessibility.	The Unified UI of the URBREATH Toolbox aims to offer a consistent and user-friendly Web interface, simplifying access to the different functionalities of the toolbox that envisages direct interaction with the end user.
BB.10 Interaction Service This BB supports the interaction of data, models and visualisations, allowing to conduct scenario-based analysis with LDTs.	The URBREATH Toolbox still does not include mechanisms implementing the functionalities envisaged by BB.10 in an automatic way. However, considering the planned next steps, the toolbox will include in its second version mechanisms based on the Apache Kafka and Apache Airflow for the execution of analyses. This represents a first stage towards BB.10.
BB.11 Message Broker This BB serves as an intermediary layer to manage the distribution of messages, ensuring that messages are correctly delivered to the right recipients.	The URBREATH Toolbox includes the macro area "Components Communication", which offers functionalities to establish intercomponent communication channels through messages distributions (via the Message Broker, i.e. Apache Kafka, which is among the tools suggested by the LDT Toolbox specifications) and the publish/subscribe capabilities (via the Context Broker, i.e. Orion-LD).
BB.12 Model Abstraction SDK/Service This BB aims to facilitating the control, orchestration, execution and composition of diverse models, enabling interoperability and managing model data.	Currently, the URBREATH Toolbox does not include this kind of capability.
BB.13 Model Catalogue This BB allows to perform the storage, versioning, and metadata tracking of machine learning models.	The URBREATH Toolbox includes a Models/Tools Catalogue that allows to menage metadata associated to models and tools (descriptions, versions, references to source

	code, manual, documentation, etc.). The Models/Tools Catalogue manages this information following DACT-AP specifications and exposes REST APIs (as well as a Web based UI) to manage models/tools metadata and search among available ones.
BB.14 Model Usage Guidelines This is not a technical BB; model usage guidelines are considered an essential component of the critical path for the development of LDTs. They form a framework that allows users to determine appropriate modelling techniques in relation to the problem that needs to be solved including data-level considerations to be considered.	This BB is addressed by the technical documentation the project already produced and will produce until its end.
BB.15 Algorithms & Models This BB describes the type of models that should be compatible in an LDT to add value and the possibility to access and integrate those models in an LDT environment allowing stakeholders to understand, plan, analyse, predict and make more data-driven decisions in order to improve urban developments.	The URBREATH includes an initial set of analysis. Among the next steps, the URBREATH Toolbox will include UI elements and process for the execution of analysis based on Apache Kafka and Apache Airflow that will allow end users (e.g. Data Scientist and Urban Planners) to configure and execute analysis and access produced results once they are ready, making them available through the Dataset Catalogue.
BB.16 Asset Registry This BB serves as an indexed catalogue of the assets involved (and available) in the Digital Twin. While the asset documentation can exist external to an LDT instance, the registry is integral for traceability and discoverability. The Asset Registry provides an encompassing overview of the available datasets, schemas, vocabularies, algorithms, and other usable assets.	The URBREATH Toolbox includes three catalogues related to datasets (ingested and generated by the toolbox, such as results from analyses), models and tools (such as the analyses available within the toolbox), and data source (from which it is possible to ingest data to feed analyses, visualizations, etc.). The Data Discovery and Sharing macro-area offers also capabilities to publish information (i.e. metadata) related to datasets, models/tools, and data sources, and for searching and discovering assets of interest among.
BB.17 Synthetic Data Generation Tool This BB aims to offer the possibility to generate artificial or simulated data that closely resemble real-world data, with similar statistical properties, distributions, and relationships as the original data.	Currently, the URBREATH Toolbox does not include this kind of capability.

BB.18 Data Storage This BB wants to offer a logical data layer to manage, integrate and transform data. This includes capabilities for metadata modelling, data access pointy, and data transformation (i.e. BB.05).	The Data Management macro area of the URBREATH Toolbox offers capabilities for data storage and data transformation and integration. Concerning data storage capabilities, this macro area includes components offering the possibility to implement and manage structured and NoSQL databases, that aligns with tools suggested by the LDT Toolbox specifications, such as PostgreSQL, MySQL, and MongDB. In addition, the Data Management macro area include components enabling timeseries DB (i.e. InfluxDB) and object storage (i.e. MinIO)

It is important to underline that the aim of the URBREATH Toolbox is not to offer a technical implementation of the LDT Toolbox Specifications, but to align with them, offering a set of technological components that enable cities to build LDT for managing the lifecycle of urban interventions concerning Nature-Based Solutions, towards future applicability of the provided technological solutions.

12 Annex D – Overview of the analysis orchestration process

This section offers an overview of the expected process for orchestrating the execution of analyses requested from the Unified UI or from the Digital Twin (i.e. 3D Map).

The process initiates from the Unified UI or from the 3D Map (according to the kind of analysis, it could be made available from one of the two or both). Here the user sets the parameters of the analysis and request its execution.

The request containing the parameters set the users and the reference to the requested analysis is submitted to the Proxy, a new component which will oversee keeping track of the requests submitted by a user, updated their status.

The proxy registers the incoming requests into the NoSQL DB (i.e. MongDB) and publishes on the message broker a message containing the parameters. The message is then intercepted by the analysis that start its execution according to the received parameters.

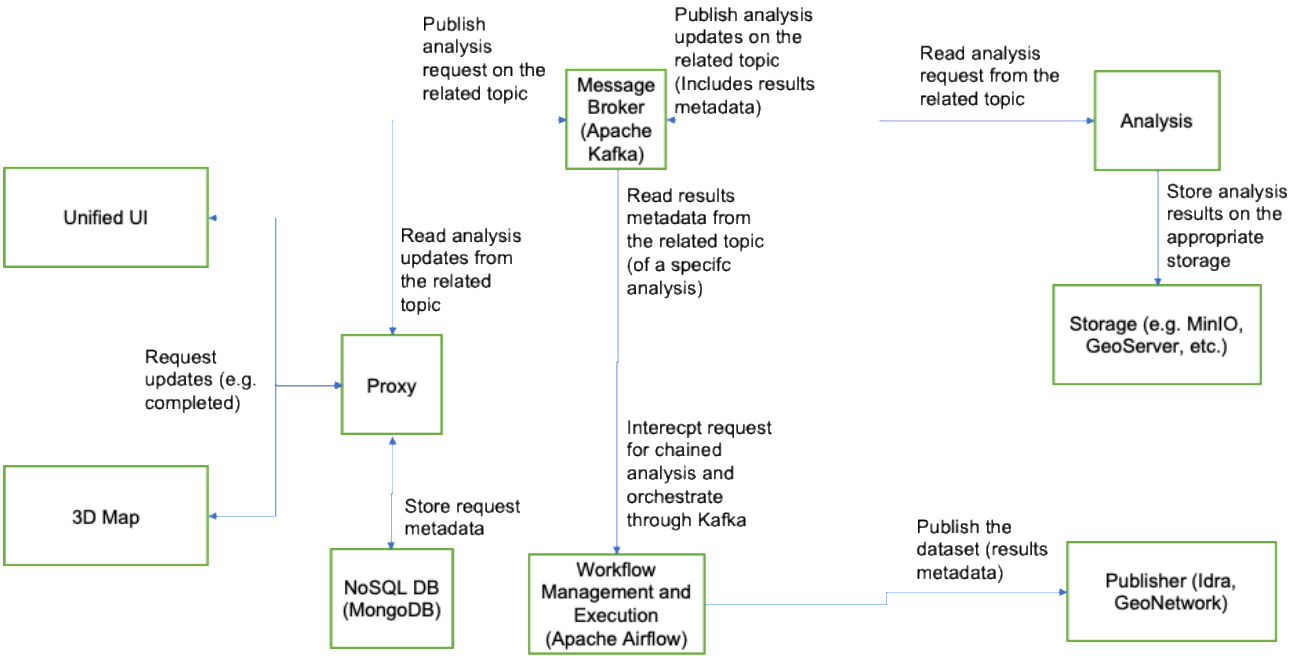
Once the analysis produces an update (e.g. status change) or the results, publishes a corresponding message on the message broker. This allows the proxy to intercept any change associated to the requested analysis and keep track of them, offering the possibility to report this information on the Unified UI or on the 3D Map.

Only when the analysis produces the expected results, it stores them into the corresponding storage (e.g. object storage, GeoServer, etc.) according to the specific type of storage that is needed.

The corresponding message published on the message broker will include also the metadata associated to the results, such as tile, description and the location where the results are stores.

This message is then intercepted also by the Workflow Management and Execution, that oversees registering the metadata through the Publisher, making the result available in the catalogue.

Figure 78: Draft design of the process for analysis orchestration



To manage information concerning incoming requests for analysis, two data models and REST APIs of the Proxy have been drafted to manage available analyses for the end users and request performed by the end users.

Table 14: Analysis draft data model

Field	Description
id	Unique identifier of the analysis. Automatically assigned.
dateCreated	Analysis creation timestamp. Automatically assigned.
dateModified	Timestamp of the last modification of the analysis. Automatically assigned.
createdBy	Reference to the user created the analysis.
endopint	Reference to the endpoint to reach the instance of the analysis.
version	Version of the analysis module
description	Text description of the instance of the analysis module.
payloadSchema	Schema representing the expected payload

Table 15: Analysis management draft REST APIs

HTTP Method	Resource	Description
GET	/analyses	Retrieve all analyses
GET	/analyses/{analysis_id}	Retrieve the details for analysis {analysis_id}
POST	/analyses	Create a new analysis
DELETE	/analyses	Remove all analyses

DELETE	/analyses/{analysis_id}	Remove analysis {analysis_id}

Table 16: Request draft data model

Field	Description
id	Unique identifier of the request. Automatically assigned.
dateCreated	Request creation timestamp. Automatically assigned.
dateModified	Timestamp of the last modification of the request. Automatically assigned.
createdBy	Reference to the user created the request.
refAnalysis	Reference to the instance of the analysis module as requested by the user.
status	Status of the analysis (PENDING; EXECUTING; ERROR; COMPLETED)
statusDescription	Text description of the status.
payload	Payload needed to execute the analysis

Table 17: Request management draft REST APIs

HTTP Method	Resource	Description
GET	/requests	Retrieve all requests
GET	/requests/{request_id}	Retrieve the details for request {request_id}
POST	/requests	Create a new request
DELETE	/requests	Remove all requests
DELETE	/requests/{request_id}	Remove request {request_id}